Scatter and Gather Operations on an Asynchronous Communication Model

Susumu Shibusawa, Hiroyuki Makino, Shigeki Nimiya and Jun-ichi Hatta
Department of Computer and Information Sciences,
Ibaraki University, Hitachi, Ibaraki 316-8511, Japan
{shibusawa, hmacky, nimi, st96067}@cis.ibaraki.ac.jp

Keywords: scatter, gather, collective communication, asynchronous communication model, communication tree.

Abstract

Collective communication such as scatter, gather, broadcast, and total exchange plays an important role on the performance of parallel and distributed processing. So far, several parallel collective communication algorithms have been studied in synchronous communication models, but it is necessary to design and analyze parallel algorithms using a more realistic communication model. This paper introduces an asynchronous communication model between processors, and evaluates the execution time of operations which include scatter, gather and some computations for data initially located at one processor. The results show that the execution time in sending data in a single direction is less than that of dual directions for asynchronous communication. This paper also introduces the logarithmic correction of the bisection communication tree, and evaluates the communication time of several instances.

1 Introduction

Collective communication such as scatter, gather, broadcast, and total exchange plays an important role in parallel and distributed processing [4], [3], [5]. Many applications such as space-time adaptive processing, collision detection of 3D virtual objects, data mining, etc. includes collective communication, and this type of communication has a large weight on the performance of applications [3], [6], [2].

So far, several parallel collective communication algorithms have been studied in synchronous communication models [9], [10]. However realistic communications between processors are usually performed asynchronously, and there is large difference between synchronous parallel algorithms and practice. From this, it is necessary to design and analyze parallel algorithms using more realistic communication models [7].

Asynchronous communication models presented so far include Log P model and BSP model [1], [11]. The Log P model has three parameters which are an upper bound on the latency, the overhead and the gap. The upper bound on the latency is a delay incurred in communicating a message containing a word from a source processor to destination processor. The overhead is the length of time that a processor is engaged in the transmission or reception of each message. The gap is the minimum time interval between consecutive message transmission or consecutive message reception at a processor.

This paper introduces another asynchronous communication model between processors, and evaluates the execution time of operations which include scatter, gather and some computations for data initially located at one processor. In the asynchronous communication model, a receiving operation at a destination processor is followed the sending operation at the source processor with a finite communication delay, and the time for sending and receiving operations is dependent on the message size. This paper also introduces two kinds of logarithmic correction for bisection communication trees, and evaluates the communication time of several instances.

The rest of the paper is organized as follows. In Section 2, we introduce an asynchronous communication model. Section 3 presents three types of communication trees. In Section 4, we present multi-communication operations of scatter and gather in an asynchronous communication model. Section 5 concludes this paper.

2 An Asynchronous Communication Model between Processors

We make the following assumptions for message communication:

Assumption 1 Exclusive transmission and exclusive reception rules hold. That is, each processor sends at most a unit of message to another processor at each step. Each processor also receives at most a unit of message from another processor at each step. □

Assumption 2 For constant message size, the time of one-to-one communication is similar between any pair of processors. □

The second assumption usually holds for parallel computer systems which have routing facilities.

We define message communication time in a synchronous communication model as follows [8]:

Definition 1 In a synchronous communication model, We define communication time t of a message with size m from one processor to another processor as follows:

$$t = \sigma + m\tau \tag{1}$$

where σ is a constant independent of the message size, and called start-up time. The symbol τ is constant in terms of message size, and called communication time of a unit-size message. □

Figure 1 shows the time dependency of message communication in the case that a message M is sent from processor P_{send} to P_{recv} in a synchronous communication model. Bold lines mean sending and receiving operations. Arrows mean the correspondence of starting-up and finishing of operations. t_{send} and t_{recv} mean the time of sending and receiving operations respectively, and t is the time required to communicate a message from P_{send} to P_{recv}. For the synchronous model, message communication time t is equal to the time of sending or receiving operation:

$$t = t_{send} = t_{recv} \tag{2}$$

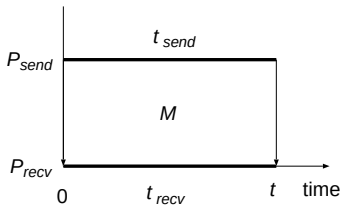

Figure 1: Time dependency of message communication for synchronous communication.

Figure 2 shows the time dependency of message communication in the case that a message M is sent from processor P_{send} to P_{recv} in an asynchronous communication model. δ is the period between the time for the source processor to begin to send a message and the time for the destination processor to begin to receive the message, which is called the start-up delay of a receiving operation. ε is the period between the time for the source processor to finish message transmission and the time for the destination processor to finish the message reception, which is called a finishing delay of receiving operation. ϕ is the period between the time for the destination processor to begin to receive message transmission and the time for the source processor to finish the message transmission, which is called the overlapping time of sending and receiving operations. In the case that the source processor finishes message transmission before the destination processor begins to receive the message as showed in Figure 2(b), the overlapping time ϕ has a negative value. In this paper, we call δ, ε and ϕ communication parameters of the asynchronous communication model.

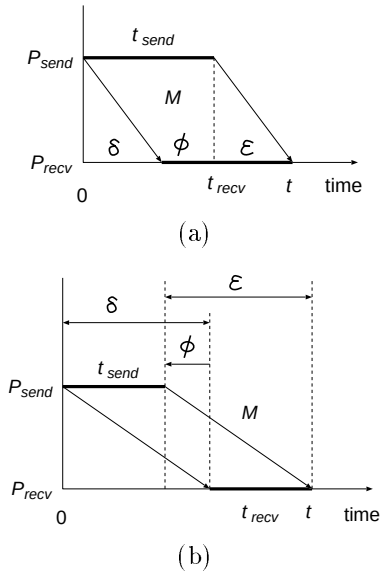

Figure 2: Time dependency of message communication for asynchronous communication. (a) Small delay. (b) Large delay.

We define the time of sending and receiving operations for the asynchronous communication model as follows:

Definition 2 In the synchronous communication model, we define the sending operation time t_{send} and receiving operation time t_{recv} of a message with size m which is sent from one processor to another as fol-

lows:

$$t_{send} = \sigma_s + m\tau_s, \quad t_{recv} = \sigma_r + m\tau_r \tag{3}$$

The symbols σ_s and σ_r are constants independent of message size, and called sending and receiving start-up times respectively. The symbols τ_s and τ_r are constant in terms of message size, and called sending and receiving operation times of a unit-size message respectively. □

We assume the communication parameter δ as follows:

Assumption 3 In a steady state where the change of a network's state is small for message communication, we assume that the start-up delay δ of receiving operation is independent of message size. □

As we can see easily from Figure 2, the following relations hold among communication parameters and message communication time.

Lemma 1 In the asynchronous communication model, the following relations hold among message communication time t, sending operation time t_{send}, receiving operation time t_{recv}, and the communication parameters δ, ϕ, ε:

$$t_{send} = \delta + \phi, \quad t_{recv} = \phi + \varepsilon \tag{4}$$

$$t = t_{send} + \varepsilon = \delta + t_{recv} = \delta + \phi + \varepsilon \tag{5}$$

When $t_{send} = t_{recv}$, the relationship $\varepsilon = \delta$ is showed. □

3 The Construction of Communication Trees

When message M_1 is sent from processor P_{i_1} to P_{i_2} and message M_2 is sent from P_{j_1} to P_{j_2}, let $L_1 = (P_{i_1}, P_{i_2})$ and $L_2 = (P_{j_1}, P_{j_2})$ be the communication paths. If the intersection of L_1 and L_2 is not empty and messages M_1 and M_2 want to use the intersection at the same time, we say that messages M_1 and M_2 have a communication overlap.

When a processor P_0 sends messages to other processors, we can construct a communication tree by connecting each pair of source and destination processors [1]. When p processors P_0, P_1, $\cdots$, P_{p-1} are connected in this order, we can construct a communication tree in a recursive manner where each pair of processors has no communication overlap.

Recursive construction of a communication tree:

1. Let node P_0 be a root of communication tree $\mathcal{T}$.
2. Choose a node P_m of communication tree $\mathcal{T}$ which consists of nodes P_i, P_{i+1}, $\cdots$, P_j, where $0 \le i \le m \le j \le p-1$, and create a directed edge (P_i, P_m) from P_i to P_m. We call node P_m a partition node of communication tree. Let $\mathcal{T}_1$ be a subtree whose root is P_m and consists of nodes P_m, P_{m+1}, $\cdots$, P_j. Let $\mathcal{T}_2$ be a subtree whose root is P_i and consists of nodes P_i, P_{i+1}, $\cdots$, P_{m-1}. Figure 3 shows a recursive construction of communication tree $\mathcal{T}$.
3. Repeat Step 2 recursively until there is no node except a root in each subtree. □

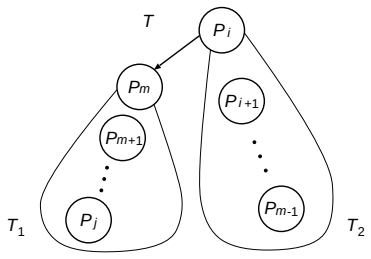

Figure 3: Recursive construction of a communication tree $\mathcal{T}$.

For communication trees constructed in the above manner, the following property holds:

Lemma 2 Let N_p be the number of p-nodes communication trees which are constructed recursively in the above manner. Then N_p is expressed as follows:

$$N_p = \sum_{m=1}^{p-1} N_m \times N_{p-m} \tag{6}$$

The lower bound of N_p is $\Omega(2^p)$. □

For a scatter operation in which a processor P_0 sends different data to each of all processors, a communication method which recursively divides the set of destination processors into two similar parts realizes the least number of steps in the synchronous communication model. We call this communication method a bisection communication. This is not true in the asynchronous communication model because the message transmission delay is accumulated by transferring a message many times. From this observation, we introduce two types of logarithmic corrections of bisection communication in the asynchronous communication model in order to reduce the maximum communication delay. For a communication tree with $q = j - i + 1$ nodes P_i, P_{i+1}, $\cdots$, P_j, we express three partition nodes P_m as follows:

1. Bisection:

$$m = \lceil \frac{q}{2} \rceil \tag{7}$$

2. Logarithmic correction:

$$m = \lfloor \frac{q}{2} + \log_2(\frac{q}{2}) \rfloor \quad (8)$$

3. Second-order logarithmic correction:

$$m = \lfloor \frac{q}{2} + \log_2 \log_2 q \rfloor \quad (9)$$

Figure 4 shows examples of communication trees for $p = 8$.

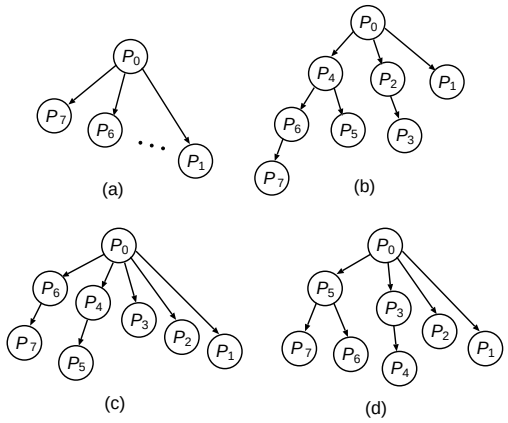

Figure 4: Examples of communication trees. (a) Sequential communication. (b) Bisection. (c) Logarithmic correction. (d) Second-order logarithmic correction.

Using the communication tree, we can derive a communication schedule diagram which shows the time dependence of interprocessor communication. Figure 5 shows an example of the second-order logarithmic correction for $p = 8$, $t_{send} = t_{recv} = \delta = 15$. In the figure, the time of sending and receiving operations and communication parameters have an arbitrary unit.

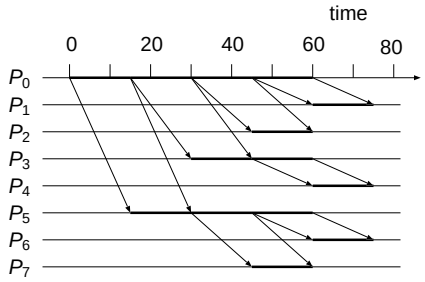

Figure 5: An example of a communication scheduling diagram for the second-order logarithmic correction.

Let $t^{(j)}_{comm}$ be the period between the time for processor P_0 to start sending a message and the time for P_j to finish receiving the message. The communication time T required for P_0 to send messages to all the other processors is expressed as follows:

$$T = \max_{1 \le j \le p-1} \{t^{(j)}_{comm}\} \quad (10)$$

Figure 6 shows an evaluation of communication time for $p = 8, 16$, $t_{send} = t_{recv} = 15$, where the time of sending and receiving operations and communication parameters have an arbitrary unit. In the figure, Seq, Binary, LC and LC2 mean the cases for sequential, bisection, logarithmic correction and second-order logarithmic correction respectively. In this figure, the communication time of two types of logarithmic correction is smaller than that of sequential and bisection communication for $15 \le \delta \le 30$. There are cases where the communication time of sequential and bisection communication is smaller than that of two types of logarithmic correction, and the construction of efficient trees with the smallest communication time is dependent to the value of parameters t_{send}, t_{recv}, δ and p. The following section provides the performance evaluation of operations which include scatter, gather and some computations for a sequential communication model and three types of bisection communication.

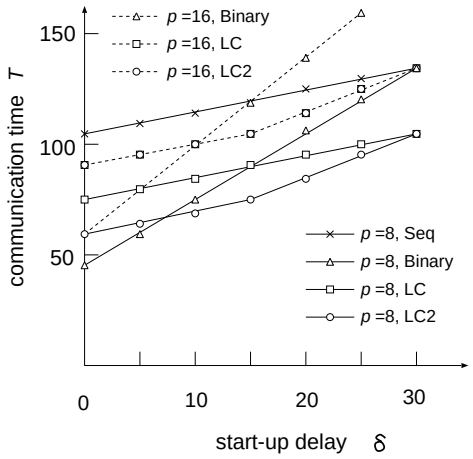

Figure 6: An evaluation of communication time.

4 Scatter and Gather of Messages

This section studies the following problem: equally divide data initially located at one processor, scatter each part of data to a different processor, perform some computation at each processor and then gather each of the results to the original processor.

4.1 Sequential Scatter and Gather

Let processor P_0 holding the initial data of size D be a communication processor, and let p processors P_1, P_2, $\cdots$, P_p be computation processors. The communication processor sends data the size of $d = D/p$ to each computation processor sequentially, which performs some computation using the data. Then each computation processor returns the computation results of size a to the original communication processor, which gathers the results sequentially. Let t_{comp} be the computation time required for each computation processor to get the results of size a. We call these scatter and gather operations a sequential scatter and a sequential gather respectively.

Lemma 3 In the synchronous communication model, the execution time of operation with sequential scatter, computation and sequential gather is expressed as follows:

$$T = 2p\sigma + p(d+a)\tau + \max\{T_{excs}, 0\} \tag{11}$$

where T_{excs} is as follows for $t_{scat} = \sigma + d\tau$:

$$T_{excs} = t_{comp} - (p-1)t_{scat}. \tag{12}$$

□

We sometimes call an operation with sequential scatter, computation and sequential gather a sequential communication operation. Figure 7 shows an example of a sequential communication operation in the asynchronous communication model. The horizontal dashed line of the figure shows an idle state of processor.

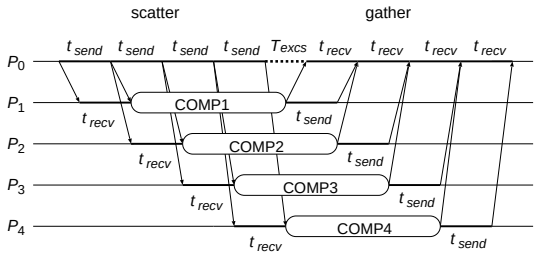

Figure 7: An example of a sequential communication operation for asynchronous communication ($T_{excs} > 0$).

Lemma 4 In the asynchronous communication model, the execution time of operation with sequential scatter, computation and sequential gather is expressed as follows:

$$T = p(\sigma_s + \sigma_r) + p(d\tau_s + a\tau_r) + \max\{T_{excs}, 0\} \tag{13}$$

where T_{excs} is as follows for $t_{send} = \sigma_s + d\tau_s$:

$$T_{excs} = t_{comp} - (p-1)t_{send} + (\delta + \varepsilon). \tag{14}$$

□

4.2 Multiple Scatter and Gather

We call the communication method that performs several scatter operations simultaneously a multiple scatter. We also call the communication method that performs several gather operations simultaneously a multiple gather. We will study multiple scatter and gather operations of bisection communication hereafter and call them multi-communication operations.

Let processor P_0 hold the initial data of size D, and each of p processors P_0, P_1, $\cdots$, P_{p-1} perform a part of some computation. Processor P_0 first sends half of the initial data to another processor. Each processor which receives data sends half of the data to another processor in parallel. For $p = 2^k$, each of p processors receives data the size of $d = D/p$, and performs some computation. Then each processor sends the computation results to another processor in parallel, and eventually P_0 gathers all of computation results.

We study multiple scatter and gather operations on a ring connection model under Assumption 2. We can show that there is no communication conflict for multiple scatter and gather operations on a ring connection model.

Lemma 5 In the synchronous communication model, the execution time of multi-communication operations for gather, computation and scatter is expressed as follows:

$$T = 2\sigma \log p + (p-1)(d+a)\tau + t_{comp} \tag{15}$$

where the base of logarithm is 2. □

Next, we will introduce three types of multi-communication operations for the asynchronous communication model according to the direction of scatter and gather. Figure 8 shows examples of two types of multi-communication operations with uni-direction in which the direction of gather is equal to that of scatter. In the first method, processor P_1 first starts gathering data, and all of the data of size pa are gathered at P_0 in $\log_2 p$ steps. In the second method, all of the data are gathered at P_{p-1} in $\log_2 p$ steps, and then they are sent to P_0.

Let t_{isend}, t_{irecv} and t_i be the time of the sending and receiving operation and communication time of data at the ith scatter step respectively. It can be easily derived that, for $1 \le i \le \log p$, $0 \le j \le p-1$, the communication time required for processor P_0 to send data to P_j at the ith scatter step is t_{isend} or

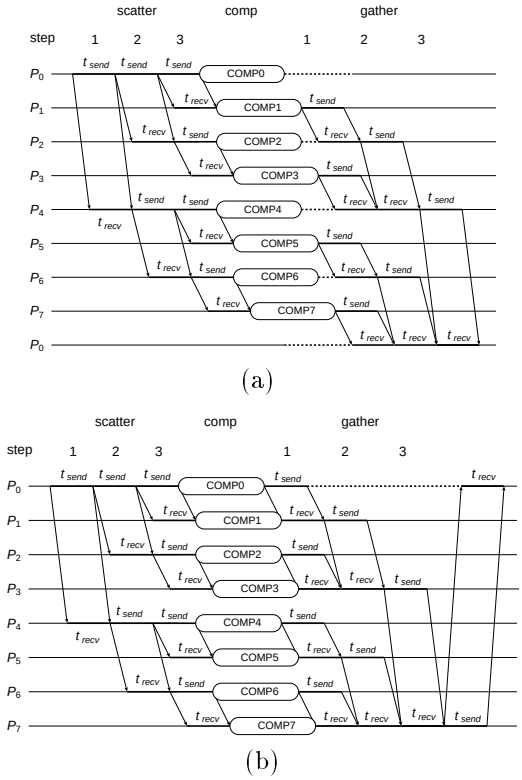

Figure 8: Asynchronous multi-communication operations with uni-direction. (a) Method 1. (b) Method 2.

t_i. From this, the following property holds for multi-communication operations:

Lemma 6 In the asynchronous communication model, let T_{dual} be the execution time of multi-communication operations in which the direction of gather is opposite to that of scatter. Let T_{uni} and T_{uni+1} also be the execution time of multi-communication operations using Methods 1 and 2 of Figure 8 respectively. Then the following properties hold in the case that $t_{isend} = t_{irecv}$:

$$T_{dual} = 2\sigma_r \log p + (p-1)(d+a)\tau_r + 2\delta \log p + t_{comp} \quad (16)$$

$$T_{uni} = 2\sigma_r \log p + (p-1)(d+a)\tau_r + (1+\log p)\delta + t_{comp} \quad (17)$$

$$T_{uni+1} = 2\sigma_r \log p + (p-1)(d+a)\tau_r + \delta \log p + (\sigma_r + pa\tau_r + \delta) + t_{comp} \quad (18)$$

where the base of logarithm is 2.

Proof: In multi-communication operations with dual directions, the total execution time is dependent on the scatter to processor P_{p-1} and gather from P_{p-1}. The time $t_{scat}^{(p-1)}$ required to scatter P_0's data to processor P_{p-1} is derived as follows:

$$t_{scat}^{(p-1)} = \sum_{i=1}^{k} [\delta + (\sigma_r + 2^{k-i} d\tau_r)] \quad (19)$$

The time $t_{scat}^{(p-1)}$ required to gather P_{p-1}'s data to P_0 is derived similarly.

In the first method with uni-direction, the total execution time of multi-communication operations is dependent on the scatter of P_0's data to P_{p-1} and the gather of P_1's data to P_0. In the case that $t_{isend} = t_{irecv}$, Eq.(17) holds for the first multi-communication method with uni-direction. In the second method with uni-direction, the total execution time of multi-communication operations is dependent on the scatter and gather of P_0's data to P_{p-1}. In the case that $t_{isend} = t_{irecv}$, Eq.(18) holds for the second multi-communication method with uni-direction. □

Theorem 1 In the asynchronous communication model, let T_{dual} be the execution time of multi-communication operations in which the direction of gather is opposite to that of scatter. Let T_{uni} and T_{uni+1} also be the execution time of multi-communication operations using Methods 1 and 2 of Figure 8 respectively. Then the first multi-communication method with uni-direction is always faster than the second multi-communication method with uni-direction and the method with dual directions for $p \geq 2$. That is,

$$T_{uni} < T_{uni+1}, \quad T_{uni} \leq T_{dual}. \quad (20)$$

□

We can also compare the first multi-communication method with the sequential communication operation.

Theorem 2 In the asynchronous communication model, let T_{seq} be the execution time of the sequential communication operation for p processors. Let T_{uni} also be the execution time of the multi-communication operation using Method 1. Then there are at most three real solutions for $T_{seq} = T_{uni}$ in the case that $t_{isend} = t_{irecv}$, $c > 0$, $p \geq 2$ and $T_{excs} \leq 0$. □

From Theorem 2, there are cases that the sequential communication operation is faster than the multi-communication operation according to the value of communication parameters.

5 Conclusion

This paper introduces multi-communication operations of scatter and gather with uni-direction in an

asynchronous communication model, and compares the performance with that for dual-directional communication and sequential communication operations. The execution time of a uni-directional communication operation is smaller than that for the dual-directional communication operation.

We are now studying efficient uni-directional communication operations in the asynchronous communication model. We are also studying efficient methods to overlap communication and computation in the asynchronous communication model. The construction of communication trees for the large number of processors and their performance evaluation are important topics. Studies about processor networks with several types of connections will also produce fruitful and realistic results.

References

[1] D. E. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser, E. Santos, R. Subramonian and T. V. Eicken, "Log P: towards a realistic model of parallel computation," *Proc. 4th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, 1-12, 1993.

[2] S. Goil and A. Choudhary, "High performance data mining using data cubes on parallel computers," *Proc. IPPS/SPDP*, 548-555, April 1998.

[3] K. Hwang, Z. Xu and M. Arakawa, "Benchmark evaluation of the IBM SP2 for parallel signal processing," *IEEE Trans. Parallel and Distrib. Sys.*, 7(5): 522-536, May 1996.

[4] S. L. Johnsson and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, 38(9): 1249-1268, Sept. 1989.

[5] J.-S. Kim and D. J. Lilja, "Performance-based path detection for interprocessor communication in distributed computing systems," *IEEE Trans. Parallel and Distrib. Sys.*, 10(3): 316-327, March 1999.

[6] Y. W. Lim and V. K. Prasanna, "Scalable portable implementations of space-time adaptive processing," *Proc. the 10th Annual Int'l Symp. on High Performance Computers*, June 1996.

[7] H. Makino and S. Shibusawa, "Performance evaluation of communication operations on a distributed memory parallel computer," *Proc. 8th Int'l Parallel Computing Workshop*, 281-288, Sept. 1998.

[8] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, 26(2): 62-76, Feb. 1993.

[9] Y. Tseng and S. K. S. Gupta, "All-to-all personalized communication in a wormhole-routed torus," *IEEE Trans. Parallel and Distrib. Sys.*, 7(5): 498-505, May 1996.

[10] Y. Tseng, T. Lin, S. K. S. Gupta and D. K. Panda, "Bandwidth-optimal complete exchange on wormhole-routed 2D/3D torus networks: a diagonal propagation approach," *IEEE Trans. Parallel and Distrib. Sys.*, 8(4): 380-396, April 1997.

[11] L. G. Valiant, "A bridging model for parallel computation," *Comm. ACM*, 33(8): 103-111, Aug. 1990.

Susumu Shibusawa received the Ph.D. degree in Information Engineering from Kyoto University, Japan, in 1991. He is now an Associate Professor in the Department of Computer and Information Sciences at Ibaraki University, Japan. His research interests include communication schemes of parallel and distributed computing systems, and their applications to image processing and virtual reality.

Hiroyuki Makino is a student at Graduate School of Ibaraki University, Japan. His research interests includes parallel programming and the performance analysis of parallel and distributed systems.

Shigeki Nimiya is a student at Graduate School of Ibaraki University, Japan. His research interests includes applications of parallel and distributed computing to image processing and virtual reality.

Jun-ichi Hatta is a student in Ibaraki University, Japan. His research interests includes the collective communication of heterogeneous distributed computing systems.