マルチメディア通信と分散処理　107－6
（２００２．３．２８）

Large-Scale Group Communication in Heterogeneous Network

Kojiro Taguchi and Makoto Takizawa
Tokyo Denki University, Japan
{tagu, taki}@takilab.k.dendai.ac.jp

We discuss a novel type of group protocol for a large number of processes which are distributed in various types of networks. A group including a large number of processes implies large computation and communication overheads for manipulating and transmitting vector clocks. In this paper, we propose a hierarchical group to causally deliver messages to a large number of processes in a group.

異種ネットワーク間における大規模グループ通信

田口 幸次郎　滝沢 誠
東京電機大学理工学部

LAN や WAN などの様々なネットワークに分散されたプロセスから構成されているグループでは、プロセスにメッセージを配送するために要する時間はそれぞれ異なる。また、多くのプロセスから構成されているグループでは、事象の順序付けにベクター時刻を用いると、計算と通信の負荷が問題となる。そこで、これらの負荷を軽減する方法として、グループを複数の副グループに分割する方法がある。本論文では、副グループ毎にベクター時刻や物理時計などの異なる事象順序化方法を用いたグループで、グループ内のプロセスにメッセージを因果配送するための方法を提案する。

1 Introduction

A group of multiple processes are cooperating to achieve some objectives in distributed applications like teleconferences. In virtual universities, students in the world can admit courses. In these applications, huge number of processes are cooperating, which are distributed in various areas like not only local area but also wide area. A *large-scale* group is a group which includes huge number of processes, i.e. hundreds of processes. A *wide-area* group is a group where processes are distributed in wide-area networks like the Internet. In a *local-area* group, processes are in a same local area network. In a *heterogeneous* group, processes are interconnected with various types of networks and are realized in different types of computers. Networks are characterized by Quality of Service (QoS) like delay time and packet loss ratio. In the homogeneous group, a communication channel between every pair of processes supports same QoS. In a heterogeneous group, some pair of channels support different QoS. Suppose there are three processes p_1, p_2 and p_3 in a group G. The processes p_1 and p_2 are connected with a local area network (LAN) in a campus, and the other process p_3 is in another campus, where the local area networks are interconnected in the Internet. Here, the group is heterogeneous and wide-area type. If p_1, p_2, and p_3 are interconnected in a same LAN, the group is homogeneous.

A group protocol supports a group of n (> 1) processes with causally/totally ordered delivery of messages [9]. In order to support the ordered delivery of messages, a vector clock [9] including n elements is used. A header length is $O(n)$. $O(n^2)$ computation and communication overheads are implied for number n of processes in a group. Even if a group of about ten processes can be realized by traditional group protocols, it is difficult, maybe impossible to support a group of hundreds of processes due to large computation and communication overheads. In order to reduce the overheads, hierarchical groups are discussed.

In traditional group protocols [1, 2], every process in a group uses a same mechanism to maintain the vector clock. We discuss a new type of *hierarchical group* (HG) communication protocol for a large-scale, wide-area, heterogeneous group of processes in this paper. Here, processes in different local areas establish a subgroup where different clocks are adopted. Subgroups are interconnected by the Internet to make a group.

In section 2, we present a system model. In section 3, we present a hierarchical group. In section 4, we discuss a the HG protocol.

2 System Model

2.1 System configuration

A system is composed of multiple processes interconnected in communication networks. A *group* of multiple processes are cooperating in order to achieve some objectives. In the one-to-one communication like one supported by TCP/IP [3] and multicast communication [4], each message is *reliably* delivered to one or more than one process, i.e. in the sending order with neither loss nor duplication of message. On the other hand, in the group communication, multiple processes first establish a *group*. Then a process sends a message to one or more than one process while receiving messages from one or more than one process in the group. The membership of the group may be dynamically changed by members' leaving and new members' joining the group [10]. In addition to supporting the reliable delivery of messages to the destination processes, messages are required to be *causally* delivered to destination processes in the group. Let $s_i(m)$ and $r_i(m)$ denote sending and receipt events of a message m in a process p_i. By using the *happens-before* relation [7], the causally precedent relation among messages is defined: a message m_1 *causally precedes* another message m_2 iff $s_i(m_1)$ *happens before* $s_j(m_2)$. A process is required to deliver a message m_1 before m_2 if m_1 causally precedes m_2. In order to causally deliver messages, the vector clock [9] is used.

Processes are interconnected in various types of personal area network (PAN) and IEEE802.11b [8], local area networks (LANs), wide area networks (WANs) like the Internet. Every pair of processes can communicate with one another through a channel supported by the network. For example, each channel is logical and realized in a connection between a pair of processes supported by TCP/IP [3]. A network is modeled to be a collection of channels. There are assumed to exist a channel $C_{ij} = \langle p_i,\ p_j \rangle$ between a pair of processes p_i and p_j.

2.2 Types of groups

A group G is *homogeneous* iff every pair of channels C_{ij} and C_{kl} supports same QoS ($Q_{ij} = Q_{kl}$) in G. In the heterogeneous group G, $Q_{ij} \neq Q_{kl}$ for some pair of channels C_{ij} and C_{kl}. If the processes are in a same LAN, the group is homogeneous since each channel supports QoS.

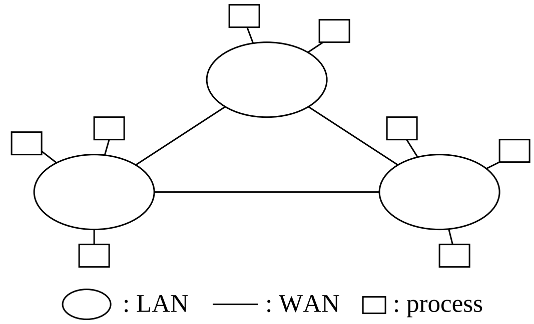

Figure 1: Heterogeneous group.

Computation and communication overheads in group communication depend on number n (> 1) of processes $p_1, \ldots, p_n$ in a group G. Here, n shows *size* of the group G denoted by $|G|$. Groups are classified into three categories, *small-scale*, *middle-scale*, and *large-scale* ones with respect to group size. The small-scale group includes about ten processes, and the middle-scale group includes about ten to hundred processes. The large-scale group has more than one hundred processes.

Groups of processes are also classified into *wide-area* and *local-area* groups. All the processes in a local-area group are interconnected in a same local area network. If some processes in a group are interconnected with the Internet, the group is a *wide-area* type. The delay time between processes is in an order of mili-seconds in the local-area group and hundreds msec in the wide-area group. Local-area groups are homogeneous. Wide-area groups are generally heterogeneous because processes are interconnected in various types of networks.

2.3 Types of group protocols

It is significant to discuss which process coordinates communication among processes in a group. One way is a *centralized* way [5, 6] where there is one controller in a group. Every process first sends a message to the controller and then the controller delivers the message to all the destination processes in the group [Figure 2]. The delivery order of messages is decided by the controller, e.g. the controller delivers messages according to the receipt order of the messages. The controller also atomically delivers messages to all the destinations, e.g. the controller retransmits a message to a destination if the destination fails to receive the message. Here, the messages are totally ordered. Another way is a *distributed* way where there is no centralized controller. Every process directly sends messages to the destination processes and directly receives messages from processes in a group. Each process makes a decision on delivery order and atomic receipt of messages by itself.

In the network, messages may be lost due to congestions and network failure. ISIS [2] takes a decentralized way where every destination process sends a receipt confirmation to the sender of a message if the process successfully receives the message. Takizawa *et al.* proposes a fully distributed way where every destination process sends a receipt confirmation to not only the sender but also all the other destinations [Figure 3]. A process can detect loss of a message on receipt of messages including receipt confirmation from other destinations. In order to reduce number of messages transmitted in the network, receipt confirmation of messages received is carried back to the other processes. In addition, every process takes *delayed confirmation* strategy. That is, a process does not send a confirmation messages as soon as the process receives a message unless there is any message to send. The process sends receipt confirmation of messages received only if the process receives some number of messages or it takes some time after most recently receiving a message. Furthermore, the *destination retransmission* is proposed [Figure 4]. Here, if a process fails to receive a message, another destination, e.g. nearest to the process, retransmits the message to the process [12]. In the other protocols, only the sender retransmits the message.

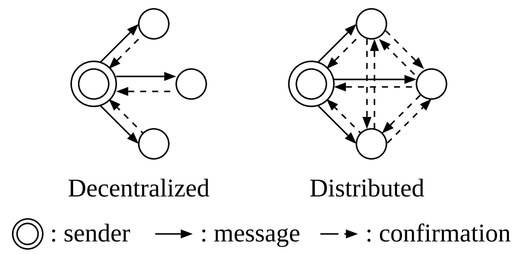

Figure 3: Confirmation.

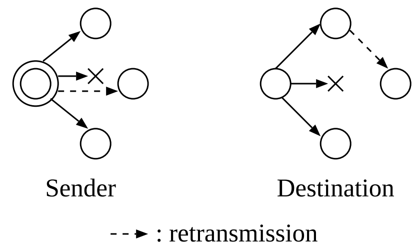

Figure 4: Retransmission.

3 Hierarchical Group

The header length of message is $O(n)$ and the computation and communication overheads are $O(n^2)$ for number n of processes in a group. In order to reduce the overheads, a group can be hierarchically structured. For example, there are one hundred processes $p_1, \ldots, p_{100}$ in a group G. Suppose a group G is decomposed into ten subgroups $G_1, \ldots, G_{10}$, each of which includes ten processes. Each subgroup G_i supports one process named a *gateway* w_i ($i = 1, \ldots, 10$). If a process in a subgroup G_i sends messages to processes in the same subgroup G_i, the messages are transmitted only in G_i. If a process p_i in G_i sends a message m to a processes p_j in another subgroup G_j ($j \neq i$), p_i first sends the message m to a gateway process w_i in G_i. Then, w_i forwards the message m to a gateway w_j of the subgroup G_j. The gateway w_j delivers the message m to the destination process p_j in G_j. Here, the header length of message exchanged in a subgroup is one tenth and the overheads of each process can be also one tenth of G. A group G is referred to as *flat* or *one-level* iff every process directly delivers messages to destination processes in G. A group G is *hierarchical* iff G is partitioned into subgroups and every process in a subgroup does not directly deliver messages to any process in another subgroup [Figure 5]. The example presented here shows a *hierarchical* group.

A group G is composed of subgroups $G_1, \ldots, G_k$ ($k \geq 1$). Each subgroup G_i is composed of processes $p_{i1}, \ldots, p_{il_i}$ ($l_i \geq 1$) and one

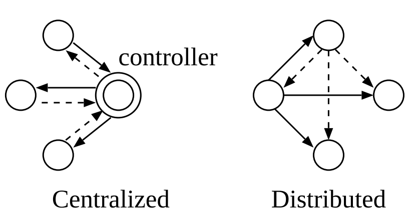

Figure 2: Transmission of message.

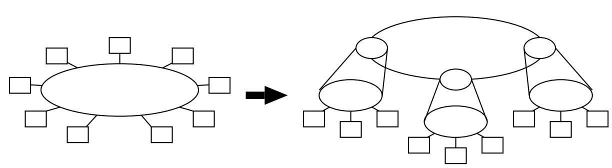

Figure 5: Hierarchical group.

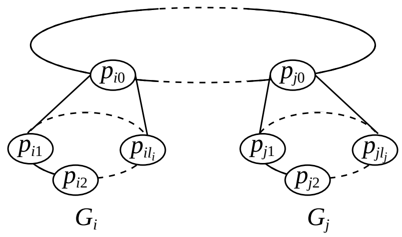

Figure 6: Hierarchical group.

gateway process p_{i0}. If G_i is a centralized (C) group, p_{i0} is a controller process of G_i. A subgroup of gateway processes $p_{i0}, \ldots, p_{m0}$ is referred to as *main subgroup* of the group G. In the main subgroup, *global* messages are exchanged by gateway processes. If a global message m_1 causally precedes another one m_2 in a main group of G, $m_1 \rightarrow_G m_2$.

In each subgroup G_i, messages can be assumed to be totally ordered by its ordering mechanism: that is, either a message m_1 *precedes* another message m_2 in G_i ($m_1 \Rightarrow_i m_2$) or $m_2 \Rightarrow_i m_1$ for every pair of messages m_1 and m_2. The precedent relation "$\Rightarrow_i$" satisfies the following property:

- $m_1 \Rightarrow_i m_2$ if m_1 causally precedes m_2 in G_i ($m_1 \rightarrow_i m_2$).

Some messages transmitted in a subgroup G_i are delivered to processes in other subgroups. Messages exchanged among subgroups are referred to as *global* messages. On the other hand, messages exchanged only in a subgroup are *local* messages. A gateway process p_{i0} takes a local message m sent by a process p_{ij} in a subgroup G_i and then forwards a global message m to destination gateways. Here, a global message m is assigned a global sequence number *gseq*. *gseq* is incremented by one each time p_{i0} forwards a local message in G_i as a global message to other subgroups. Here, $m_1.gseq < m_2.gseq$ if $m_1 \Rightarrow_i m_2$. Each gateway process p_{i0} maintains a vector $V = \langle V_1, \ldots, V_k \rangle$ for number k of subgroups. Here, V_i shows a global sequence number *gseq* of a gateway process p_{i0} ($i = 1, \ldots, k$).

Suppose a group G includes a pair of subgroups G_i and G_j. Processes p_{i0} and p_{j0} are gateway processes of subgroups G_i and G_j, respectively. A process p_{is} in G_i sends a message m_1 to p_{jt} in G_j. A process p_{jt} sends a message m_2 before receiving m_1 and a message m_3 after receiving m_1 as shown in Figure 7. Here, m_1 causally precedes m_2 ($m_1 \rightarrow m_2$) but m_1 and m_2 are causally concurrent ($m_1 \parallel m_2$). In a main subgroup of G, gateway processes exchange messages by taking

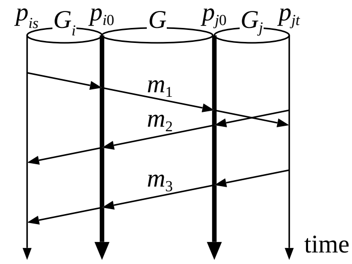

Figure 7: Causal delivery in hierarchical group.

usage of vector clock. The process p_{j0} sends m_2 to p_{i0} after receiving m_1. Hence, m_1 causally precedes m_2 ($m_1 \rightarrow_G m_2$) in the main group of G. $m_1 \rightarrow_G m_2$ if $m_1 \rightarrow m_2$. However, $m_1 \rightarrow m_2$ does not necessarily hold even if $m_1 \rightarrow_G m_2$. We have to discuss a mechanism for not causally ordering a pair of messages m_1 and m_2 in a main subgroup of G unless $m_1 \rightarrow m_2$.

4 Protocol

We discuss a hierarchical group (HG) protocol for a hierarchical group G composed of subgroups $G_1, \ldots, G_n$ ($n \geq 1$). Each subgroup G_i includes a gateway process p_{i0} and local processes $p_{l1}, \ldots, p_{il_i}$ ($l_i \geq 1$). Each subgroup G_i is assigned a unique subgroup identifier. Each global message M exchanged among gateway processes includes following fields:

$M.SG$ = sender subgroup.
$M.DG$ = set of destination subgroups.
$M.VC$ = vector $[VC_1, \ldots, VC_n]$.
$M.DATA$ = data.

Each local message m exchanged among processes in a subgroup G_i includes following fields:

$m.sp$ = source process.
$m.dp$ = set of destination processes.
$m.SG$ = source subgroup G_i.
$m.DG$ = set of destination subgroups.
$m.vc$ = vector $\langle vc_1, \ldots, vc_n \rangle$.
$m.data$ = data.

Each gateway process p_{i0} of G_i is not only a local process of G_i but also exchanges global messages with other gateway processes. p_{i0} manipulates a *global* vector $VC = [VC_1, \ldots, VC_n]$. Each local process p_{ij} in G_i manipulates a *local* vector $vc = \langle vc_1, \ldots, vc_n \rangle$ ($j = 0, 1, \ldots, l_i$). Here, n shows the number of subgroups in G. Initially, each value of VC and vc is 0 in each process.

First, suppose p_{is} in a subgroup G_i sends a local message m to p_{jt} in another subgroup G_j. Here, $m.sp = p_{is}$, $m.SG = G_i$, $p_{jt} \in m.dp$, and

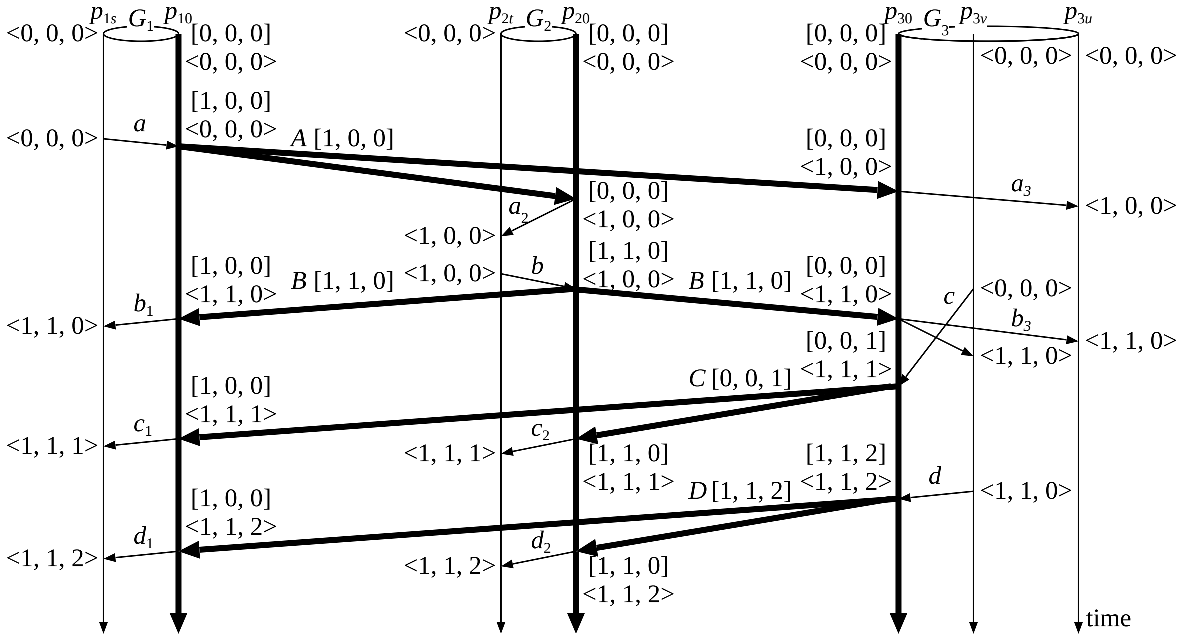

Figure 8: Communication among subgroups.

$G_j \in m.DG$. The process p_{is} sends m to a gateway process p_{i0} where $m.vc := vc$. It is noted that the local vector vc of p_{is} is not updated on sending a local message while the vector clock is incremented on sending a message.

Then, the gateway process p_{i0} receives the local message m. The global vector VC in p_{i0} is manipulated as $VC_i := VC_i + 1$; $VC_k := m.vc_k$ ($k = 1, \ldots, n,\ k \neq i$). Then, a global message M is created for the local message m where $M.VC := VC$, $M.SG := m.SG$, $M.DG := m.DG$, and $M.DATA = m$. The gateway process p_{i0} sends the global message M to each destination gateway p_{j0} where $G_j \in M.DG$.

Next, a gateway process p_{j0} receives a global message M from G_i. Here, the global vector VC in p_{j0} is manipulated as $vc_j := VC_j$; $vc_k := \max(vc_k,\ M.VC_k)$ $(k = 1, \ldots, n,\ k \neq j)$. p_{j0} creates a local message m from the global message M and then forwards m to destination processes in G_j. Here, $m := M.DATA$ and $m.vc := vc$. Each gateway process has a pair of local vector vc and global vector VC while a local process only manipulates a local vector vc.

A local process p_{jt} receives a local message m from the gateway process p_{j0}. Here, the local vector vc in p_{jt} is manipulated as $vc_k := \max(vc_k,\ m.vc_k)$ $(k = 1, \ldots, n,\ k \neq j)$. A local process p_{is} sends a local message m to not only processes in other subgroups but also processes in a same subgroup G_i. Suppose that p_{it} receives a local message m from p_{is}. The local vector vc in p_{it} is manipulated as $vc_k := \max(vc_k,\ m.vc_k)$ $(k = 1, \ldots, n,\ k \neq i)$.

Figure 8 shows a group composed of three subgroups G_1, G_2, and G_3. p_{10}, p_{20}, and p_{30} show gateway processes of the subgroups G_1, G_2, and G_3, respectively. $[VC_1,\ VC_2,\ VC_3]$ and $\langle vc_1,\ vc_2,\ vc_3 \rangle$ indicate instances of global and local vectors, respectively, in each process. Initially all the values in the vectors are 0. First, a process p_{1s} in the subgroup G_1 sends a local message a to a pair of processes p_{2t} and p_{3u} in subgroups G_2 and G_3, respectively. Here, $a.vc = \langle 0,\ 0,\ 0 \rangle$. The local message a is sent to the gateway process p_{10}. p_{10} creates a global message A from a. Here, VC_1 is incremented by one and $A.VC = [1,\ 0,\ 0]$. The gateway process p_{10} sends A to a pair of gateway processes p_{20} and p_{30}.

The local vectors vc in the gateway process p_{20} and p_{30} are changed to $\langle 1,\ 0,\ 0 \rangle$. The gateway process p_{20} sends a local message a_2 for the global message A to a local destination process p_{2t}. On receipt of a_2, vc is changed to $\langle 1,\ 0,\ 0 \rangle$ in p_{2t}. Then, p_{2t} sends a local message b with $vc = \langle 1,\ 0,\ 0 \rangle$ to the gateway process p_{20}. The second element of VC is incremented by one, i.e. $VC = [0,\ 1,\ 0]$ in p_{20}. VC is changed to $[1,\ 1,\ 0]$ by taking $\max([0,\ 1,\ 0],\ b.vc = \langle 1,\ 0,\ 0 \rangle)$. The gateway process p_{20} creates a global message B and then sends B to p_{10} and p_{30}. p_{10} forwards a local message b_1 of a global message B for the local message b with $b.vc = \langle 1,\ 1,\ 0 \rangle$. Here, since $a.vc < b_1.vc$, a causally precedes b.

In the subgroup G_3, a process p_{3v} sends a local message c with $c.vc = \langle 0,\ 0,\ 0 \rangle$ before receiving a local message b_3 with $b_3.vc = \langle 1,\ 1,\ 0 \rangle$. The gateway process p_{30} sends a global message C for the local message c after receiving the global message B. According to the traditional definition of the causality, B causally precedes C since p_{30} sends C after receiving B. However, since c is sent before b_3 is received by p_{3v}, a pair of global messages B and C must be causally concurrent. The global message B carries a global vector $VC = [1,\ 1,\ 0]$ while the global message C carries $[0,\ 0,\ 1]$. A destination process p_{1s} receives a local message c_1 of C where $c_1.vc = \langle 1,\ 1,\ 1 \rangle$. The local message b_1 of B carries the local vector $b_1.vc = \langle 1,\ 1,\ 0 \rangle$. Here, local vectors $\langle 1,\ 1,\ 1 \rangle$ and $\langle 1,\ 1,\ 0 \rangle$ are not comparable. Here, b_1 and c_1 are causally concurrent in the process p_{1s}.

A pair of messages m_1 and m_2 received are causally ordered in a local process p_{it} in a subgroup G_i according to a following ordering rule:

[Ordering rule] A message m_1 causally precedes another message m_2 in a subgroup G_i ($m_1 \rightarrow_{G_i} m_2$) if $m_1.vc < m_2.vc$. $\square$

[Theorem] A message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$) iff m_1 causally precedes m_2 in a subgroup G_i ($m_1 \rightarrow_{G_i} m_2$). $\square$

Even if a message m_1 causally precedes another message m_2 in a main group ($m_1 \rightarrow_G m_2$), the causality "$m_1 \rightarrow m_2$" does not necessarily hold. However, if $m_1 \rightarrow_{G_i} m_2$ in a subgroup G_i by the HG protocol, m_1 causally precedes m_2.

5 Concluding Remarks

We discussed the group protocol for large-scale, wide-area, heterogeneous group of processes. A group is hierarchically structured in a set of subgroups of processes.

References

[1] Ahamad, M., Raynal, M., and Thia-Kime, G., "An Adaptive Protocol for Implementing Causally Consistent Distributed Services," *Proc. of IEEE ICDCS-18*, 1998, pp.86–93.

[2] Birman, K., "Lightweight Causal and Atomic Group Multicast," *ACM Trans. on Computer Systems,* 1991, pp.272–290.

[3] Defense Communications Agency, "DDN Protocol Handbook," 1985, Vol.1–3, NIC 50004-50005.

[4] Deering, S., "Host Groups: A Multicast Extension to the Internet Protocol," *RFC* 966, 1985.

[5] Hofmann, M., Braun, T., and Carle, G., "Multicast communication in large scale networks," *Proc. of IEEE HPCS-3*, 1995.

[6] Kaashoek M. F. and Tanenbaum A. S., "An Evaluation of the Amoeba Group Communication System," *Proc. of IEEE ICDCS-16*, 1996, pp.436–447.

[7] Lamport, L., "Time, Clocks, and the Ordering of Events in a Distributed System," *CACM*, Vol.21, No.7, 1978, pp.558–565.

[8] LAN MAN Standards Committee of the IEEE Computer Society, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band.," Sept. 2001.

[9] Mattern, F., "Virtual Time and Global States of Distributed Systems," *Parallel and Distributed Algorithms* (Cosnard, M. and Quinton, P. eds.), *North-Holland*, 1989, pp.215–226.

[10] Reiter, M. K., "The Rampart Toolkit for Building High-Integrity Services," *Theory and Practice in Distributed Systems, LNCS 938*, Springer-Verlag, 1995, pp.99–110.

[11] Shimamura, K., Tanaka, K., and Takizawa, M., "Causally Ordered Delivery of Multimedia Objects," to apear in *Computer Communications Journal*, 2002.

[12] Tachikawa, T., Higaki, H., and Takizawa, M., "Group Communication Protocol for Real-time Applications," *Proc. of IEEE ICDCS-18*, 1998, pp.40–47.

[13] Tachikawa, T., Higaki, H., and Takizawa, M., "Δ-Causality and ε-Delivery for Wide-Area Group Communications," *Computer Communications Journal*, *Vol.* 23, *No.* 1, 13–21, 2000.

[14] Takizawa, M., Takamura, M., and Nakamura, A., "Group Communication Protocol for Large Group," *Proc. of the 18th IEEE Conf. on Local Computer Networks (LCN)*, 1993, pp.310–319.