Protocol for a Two-Layered Group

Kojiro Taguchi, Tomoya Enokido, and Makoto Takizawa
Dept. of Computers and Systems Engineering
Tokyo Denki University, Japan
E-mail {tagu, eno, taki}@takilab.k.dendai.ac.jp

A group including a larger number of processes implies larger computation and communication overheads to manipulate and transmit messages. In this paper, we discuss a group which is composed of subgroups of processes to reduce the overheads. Each subgroup has a gateway process which communicates with the other gateway processes. We propose a protocol to causally deliver messages to processes in a group by using a vector of message sequence numbers whose size is the number of subgroups, smaller than number of processes. We evaluate the protocol.

2階層グループのためのグループ通信プロトコル

田口 幸次郎　榎戸 智也　滝沢 誠
東京電機大学理工学部

分散システムでは複数のプロセスがグループを構成し、メッセージの送受信により協調動作を行う。このようなグループ内のプロセス間通信を実現するグループ通信プロトコルの多くは、ベクタ時刻という論理時間を用いて因果順序に基づいてメッセージの順序付けを行っている。しかし、ベクタ時刻はグループ内のプロセス数の要素を持つベクトルを、メッセージに付加する。そのため、多くのプロセスから構成されているグループでは、計算と通信の負荷が問題となる。そこで本論文では、グループを2階層のグループとし、グループ内のプロセスに因果順序に基づいてメッセージの配送を行うための方式の提案および評価を行っている。

1 Introduction

A group of multiple processes is cooperating to achieve some objectives in distributed applications like teleconferences. In these applications, huge number of processes are cooperating, which are distributed in not only local area but also wide area. A *large-scale* group is a group which includes hundreds of processes. Each communication channel between processes may not supports same Quality of Service (QoS). A *wide-area* group is a group where processes are distributed in wide-area networks like the Internet. Tachikawa and Takizawa [9,10] discuss protocols for wide-area groups which adopt fully distributed control and destination retransmission.

A group communication protocol supports a group of n (> 1) processes with causally/totally ordered delivery of messages [1,6]. In order to support the ordered delivery of messages, a *vector clock* [1,6] including n elements is used assuming that underlying networks are reliable. Here, a header length of messages is $O(n)$ for number n of processes in the group. $O(n^2)$ computation and communication overheads are implied. Even if a group of tens 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 [3, 11]. Papers [2,3] discuss how to multicast messages in tree routings but do not discuss ordered delivery of messages. Takamura and Takizawa [11] discuss how to support the causally ordered delivery in a hierarchical group by using the vector clock but the the vector size is the total number of processes. In this paper, a group is composed of subgroups each of which includes processes in a local area. Subgroups are interconnected by the Internet. We discuss a *two-layered group* (*TG*) protocol for a large-scale and wide-area group of processes. Messages are ordered by using a type of vector clock whose size is the number of subgroups, smaller than the total number of processes. Furthermore, we assume underlying networks are less reliable, i.e. message may be lost and delivered out of order. The TG protocol supports the causally ordered delivery of messages while detecting and recovering from message loss.

In section 2, we present a system model. In section 3, we discuss the causally ordered delivery in a two-layered group. In section 4, we discuss the TG protocol. In section 5, we evaluate the TG protocol in terms of delay time.

2 System Model

2.1 Groups

A *group* of multiple processes is cooperating in order to achieve some objectives in a distributed system. In the one-to-one communication and multicast communication [2], each message is *reliably* delivered to one or more than one process. On the other hand, multiple processes first establish a *group* in the group communication. Then, a process sends a message to multiple processes while receiving messages from multiple processes in the group. Here, a message m_1 *causally precedes* another message m_2 ($m_1 \rightarrow m_2$) iff a sending event of m_1 *happens before* [5] a sending event of m_2 [1]. A process is required to deliver a message m_1 before m_2 if $m_1 \rightarrow m_2$.

Due to the computation and communication overheads $O(n^2)$ for number n of processes in a group, it is difficult to support a larger group with the group communication service. In order to reduce the overheads, a group G is composed of disjointed subgroups $G_i, \ldots, G_k$. Each subgroup G_i is composed of processes and a *gateway* process p_{i0}. If a process p_i in a subgroup G_i sends a message m to destination processes in another subgroup G_j ($j \neq i$), p_i first sends m to a gateway process p_{i0} in G_i. Then, p_{i0} forwards m to a gateway process p_{j0} of the destination subgroup G_j. The gateway process p_{j0} delivers m to the destination processes in G_j. Such a group as G is referred to as *two-layered* [Figure 1]. A group is *flat* iff every pair of processes in the group directly exchange messages. For example, processes in a subgroup are interconnected in a local area network. A pair of gateway processes are interconnected in the Internet.

It is significant to discuss which process coordinates communication among processes in a group. In a *centralized* way [3,4], 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. The delivery order of messages is decided by the controller. Thus, the messages easily can be totally ordered. In a *distributed* way, 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, e.g. by using the vector clock [6]. ISIS [1] takes a *decentralized* way where every destination process sends a receipt confirmation to the sender of a message in a reliable underlying network. Takizawa *et al.* [7,8,10] take a *fully distributed* approach where every destination process sends a receipt confirmation to not only the sender but also all the other destinations in less-reliable networks.

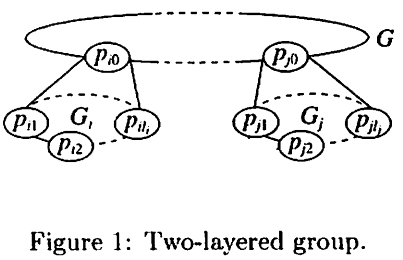

Figure 1: Two-layered group.

2.2 Confirmation vector

For a group G of n (> 1) processes $p_1, \ldots, p_n$, a vector V is in a form $\langle V_1, \ldots, V_n \rangle$ [6]. Every process p_i has a vector $V = \langle V_1, \ldots, V_n \rangle$ where each element V_j is initially 0 ($j = 1, \ldots, n$). Each time a process p_i sends a message m, V_i is incremented by one. Then, the message m carries the vector V ($m.V$) of the sender process p_i. On receipt of a message m from another process, $V_k := \max(V_k, m.V_k)$ ($k = 1, \ldots, n$, $k \neq i$) in a process. Here, for a pair of vectors $A = \langle A_1, \ldots, A_n \rangle$ and $B = \langle B_1, \ldots, B_n \rangle$, $A \leq B$ iff $A_j \leq B_j$ ($j = 1, \ldots, n$). A message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$) iff $m_1.V \leq m_2.V$. m_1 is *causally concurrent* with m_2 ($m_1 \parallel m_2$) iff neither $m_1 \rightarrow m_2$ nor $m_2 \rightarrow m_1$.

The confirmation vector RSQ of message sequence numbers is used to detect message loss in protocols [7,8]. A sequence number seq is incremented by one in a process p_i each time p_i sends a message. The process p_i has a variable rsq_j which shows a sequence number seq of message which p_i expects to receive next from a process p_j ($j = 1, \ldots, n$). Each message m carries the confirmation $m.RSQ (= \langle m.rsq_1, \ldots, m.rsq_n \rangle)$. On receipt of a message m from a process p_j, a process p_i *accepts* m if $rsq_j = m.seq$ and then $rsq_j := rsq_j + 1$. The confirmations $m.rsq_1, \ldots, m.rsq_n$ are stored in a matrix ACK as $ACK_{jk} := m.rsq_k (k = 1, \ldots, n)$. A message m received from a process p_j is *pre-acknowledged* in a process p_i if $m.seq < \min(ACK_{1j}, \ldots, ACK_{nj})$, i.e. p_i knows that every other process has accepted m. If every message is destined to all the processes, $m_1 \rightarrow m_2$ iff $m_1.RSQ < m_2.RSQ$ [8].

A process p_i can deliver a pre-acknowledged message m if every message causally preceding m is delivered and p_i receives from every process a pre-acknowledged message causally preceded by m. Here, the message m is *acknowledged* in a process p_i. The process p_i is sure that the message m is pre-acknowledged in every process, i.e. every process knows that every other process accepts m.

On receipt of a message m from a process p_j, if $rsq_j < m.seq$, the process p_i finds a message gap, i.e. p_i loses a message m' from p_j where $rsq_j \leq m'.seq < m.seq$. Next, suppose a process p_k sends a message m_1 to a pair of processes p_i and p_j but p_i fails to receive m_1. After receiving m_1, p_j sends a message m_2 to p_i where $m_2.rsq_k = m_1.seq + 1$. The process p_i receives m_2 where $rsq_k < m_2.rsq_k$ and finds that p_i has not received m_1 from p_k. Thus, p_i finds loss of a message m from another process p_k on receipt of a message m' from p_j if $rsq_k \leq m.seq < m'.rsq_k$ ($k \neq j$).

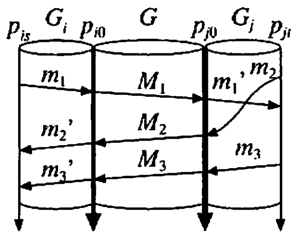

Figure 2: Two-layered group (TG).

3 Causally Ordered Delivery in Two-layered Group

A two-layered group (TG) G is composed of multiple subgroups $G_1, \ldots, G_k$ $(k > 1)$. Each subgroup G_i includes processes $p_{i1}, \ldots, p_{il_i}$ $(l_i > 1)$ and one gateway process p_{i0}. Processes and messages transmitted in a subgroup are referred to as *local*. A *main subgroup* is composed of gateway processes $p_{10}, \ldots, p_{k0}$ where *global messages* are exchanged. If a local message m is destined to a process in another subgroup, m is an *outgoing* local message. An outgoing local message m sent in a subgroup G_i is changed to a global message M. Then, the global message M is transmitted in a main subgroup and then is changed to a local massage m_j in a destination subgroup G_j. Here, m and m_j are *source* and *destination* local messages of a global message M, $sl(M)$ and $dl_j(M)$, respectively. A capital character like M shows a global message for a local message m. Let $dl_j(m)$ denote a destination local message of a source local message m in G_j. Let $sl(m)$ be a source local message of a destination local message m. Let $g(m)$ denote a global message of a local message m. A notation "$M_1 \rightarrow_G M_2$" shows that a global message M_1 causally precedes M_2 in a main subgroup of G, i.e. among the gateway processes. A notation "$m_1 \rightarrow_i m_2$" indicates that a local message m_1 causally precedes m_2 in G_i.

[Definition] A local message m_1 *causally precedes* another local message m_2 $(m_1 \rightarrow m_2)$ iff $sl(m_1) \rightarrow_i sl(m_2)$, $dl_i(m_1) \rightarrow_i sl(m_2)$, or $m_1 \rightarrow m_3 \rightarrow m_2$ for some local message m_3. □

[Theorem 1] $g(m_1) \rightarrow_G g(m_2)$ if $m_1 \rightarrow m_2$. □

Suppose a group G includes a pair of subgroups G_i and G_j whose gateway processes are p_{i0} and p_{j0}, respectively. A process p_{is} in G_i sends a local message m_1 to p_{jt} in G_j. The process p_{jt} sends a local message m_2 before receiving a destination local message $m_1'(= dl_j(m_1))$ and a local message m_3 after receiving m_1' as shown in Figure 2. Since p_{j0} sends M_2 to p_{i0} after receiving M_1, $M_1 \rightarrow_G M_2$. However, $m_1 \parallel m_2$. "$M_1 \rightarrow_G M_2$" if "$m_1 \rightarrow m_2$" from Theorem 1. 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 ordering a pair of global messages $M_1(= g(m_1))$ and $M_2(= g(m_2))$ unless "$m_1 \rightarrow m_2$" holds.

4 TG Protocol

We discuss a *broadcast two-layered group* (B-TG) G where processes send messages to all the processes. We assume that networks are less reliable, i.e. messages may be lost due to communication fault like congestion and unexpected delay. We discuss a basic data transmission procedure to causally order messages in a two-layered group (TG) G. Each local message m includes following fields:

- $m.seq$ = local sequence number.
- $m.sg$ = source subgroup G_i.
- $m.sp$ = source process in $m.sg$.
- $m.rsq$ = vector $\langle rsq_0, rsq_1, \ldots, rsq_{l_i} \rangle$.
- $m.RSQ$ = vector $[RSQ_1, \ldots, RSQ_k]$.
- $m.data$ = data.

Each global message M includes following fields:

- $M.GSQ$ = global sequence number.
- $M.SG$ = sender subgroup.
- $M.SP$ = source process in $M.SG$.
- $M.RSQ$ = vector $[RSQ_1, \ldots, RSQ_k]$.
- $M.DATA$ = data.

Each local process p_{ij} in G_i has following variables:

- seq = local sequence number.
- $rsq = \langle rsq_0, rsq_1, \ldots, rsq_{l_i} \rangle$.
- $RSQ = [RSQ_1, \ldots, RSQ_k]$.
- $ack = l_i \times l_i$ matrix.

Each gateway process p_{i0} has following variables:

- GSQ = global sequence number.
- $ACK = k \times k$ matrix.

GSQ, seq, and each element in vectors RSQ and rsq are initially 1 in every process.

First, a local process p_{is} in G_i sends a source local message m as follows:

$m.sp := p_{is}$. $m.sg := G_i$.
$m.seq := seq$. $seq := seq + 1$. $m.rsq_s := seq$.
$m.rsq_u := rsq_u$ $(u = 0, 1, \ldots, l_i, \ u \neq s)$.
$RSQ_i := RSQ_i + 1$. $m.RSQ := RSQ$.

Then, the gateway process p_{i0} receives the outgoing local message m from p_{is} in G_i. Here, variables are manipulated in p_{i0} as follows:

$rsq_s := rsq_s + 1$.
$ack_{su} := m.rsq_u$ $(u = 0, 1, \ldots, l_i)$.

Then, p_{i0} sends all the gateway processes a global message $M(= g(m))$ which is created from m as follows:

$M.SG := m.sg$. $M.SP := m.sp$.
$M.GSQ := GSQ$. $GSQ := GSQ + 1$.
$M.RSQ_h := m.RSQ_h$ $(h = 1, \ldots, k, \ h \neq i)$.
$M.RSQ_i := GSQ$. $M.DATA := m.data$.

Next, a gateway process p_{j0} in a subgroup G_j receives a global message M from G_i. Here, variables

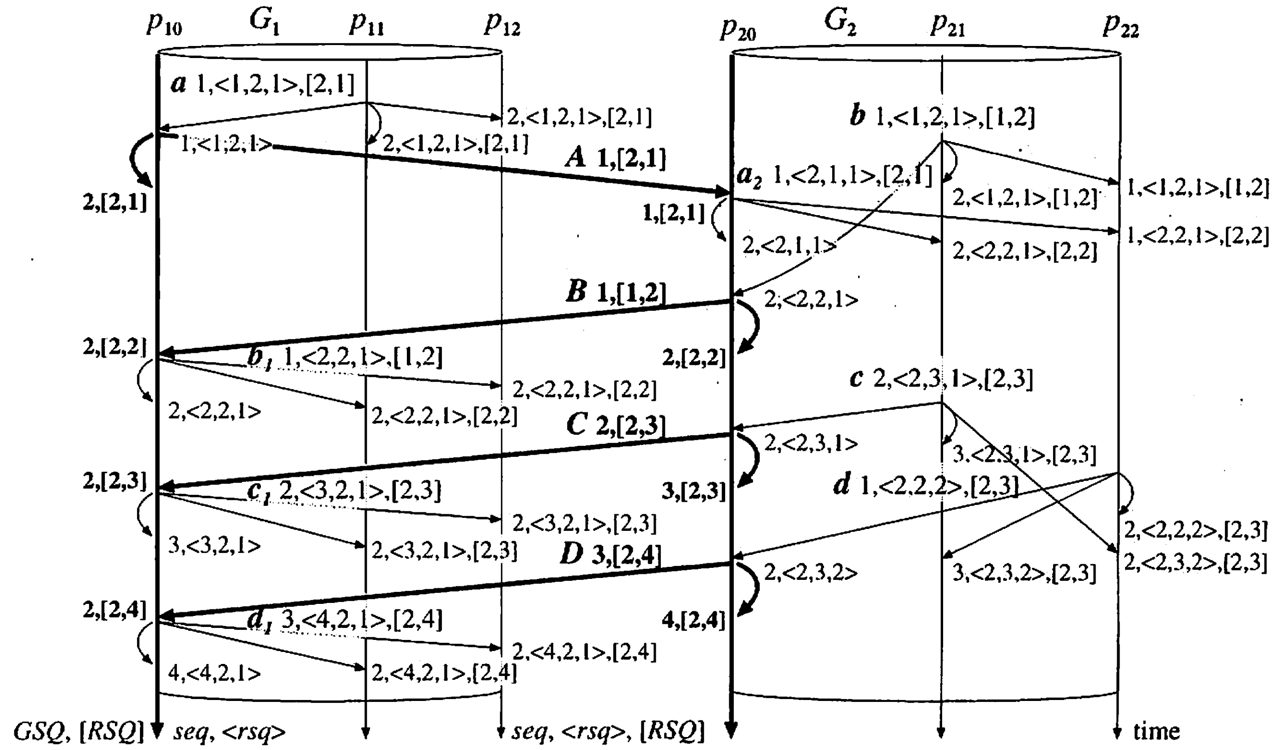

Figure 3: Communication among subgroups G_1 and G_2.

are manipulated in the gateway process p_{j0} as follows:

$RSQ_i := RSQ_i + 1$.
$ACK_{ih} := M.RSQ_h$ $(h = 1, \dots, k)$.

p_{j0} sends all the processes in G_j a destination local message $m_j (= dl(M))$ created from M as follows:

$m_j.sp := M.SP$. $m_j.sg := M.SG$.
$m_j.seq := seq$. $seq := seq{+}1$. $m_j.rsq_0 := seq$.
$m_j.rsq_u := rsq_u$ $(u = 1, \dots, l_j)$.
$m_j.RSQ := M.RSQ$. $m_j.data := M.DATA$.

A local process p_{jt} receives m_j from p_{j0}:

$rsq_0 := rsq_0 + 1$.
$ack_{0u} := m_j.rsq_u$ $(u = 0, 1, \dots, l_j)$.
$RSQ_h := \max(RSQ_h, m_j.RSQ_h)$ $(h = 1, \dots, k)$.

If p_{it} receives a local message m from p_{is} in a same subgroup G_i, variables are manipulated in p_{it} as follows:

$rsq_s := rsq_s + 1$.
$ack_{su} := m.rsq_u$ $(u = 0, 1, \dots, l_i)$.
$RSQ_h := \max(RSQ_h, m.RSQ_h)$ $(h = 1, \dots, k)$.

[Ordering rule 1] A local message m_1 *precedes* another local message m_2 in a subgroup G_i $(m_1 \Rightarrow_i m_2)$ if $m_1.rsq < m_2.rsq$ and $m_1.RSQ < m_2.RSQ$. □

[Theorem 2] If a local message m_1 causally precedes another local message m_2 $(m_1 \rightarrow m_2)$, m_1 precedes m_2 in a subgroup G_i $(m_1 \Rightarrow_i m_2)$ by the ordering rule 1. □

Global messages are causally ordered in a gateway process according to a following ordering rule:

[Ordering rule 2] A global message M_1 *precedes* another global message M_2 in a main subgroup $(M_1 \Rightarrow_G M_2)$ if $M_1.RSQ < M_2.RSQ$. □

[Theorem 3] If a global message M_1 causally precedes another global message M_2 in a main subgroup $(M_1 \rightarrow_G M_2)$, $M_1 \Rightarrow_G M_2$ by the ordering rule 2. □

Even if a global message M_1 causally precedes another global message M_2 in a main subgroup $(M_1 \rightarrow_G M_2)$, "$m_1 \rightarrow m_2$" does not necessarily hold for local messages m_1 and m_2 of M_1 and M_2, respectively. Suppose a gateway process p_{i0} receives outgoing local messages m_1 and m_2 from local processes p_{i1} and p_{i2} in a subgroup G_i, respectively. The gateway process p_{i0} creates global messages M_1 and M_2 from m_1 and m_2, respectively. p_{i0} sends M_1 before M_2 if m_1 causally precedes m_2. Here, suppose m_1 and m_2 are causally concurrent $(m_1 \parallel_i m_2)$. In the TG protocol, each time the gateway process p_{i0} sends a global message M, $M.RSQ_i := GSQ$ and $GSQ := GSQ + 1$. If p_{i0} receives m_1 before m_2, $M_1.RSQ < M_2.RSQ$, i.e. M_1 precedes M_2. Thus, for a pair of local messages m_1 and m_2 sent in a same subgroup, $g(m_1)$ may precede $g(m_2)$ even if $m_1 \parallel m_2$.

[Theorem 4] If m_1 causally precedes m_2 $(m_1 \rightarrow m_2)$ and $m_1.sg \neq m_2.sg$, a global message $g(m_1)$ precedes another global message $g(m_2)$ by the ordering rules.

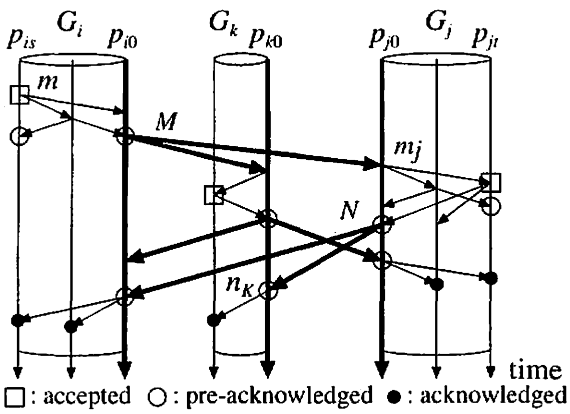

Figure 4: Data transmission.

□

[Example] Figure 3 shows a group G composed of two subgroups G_1 and G_2 where p_{10} and p_{20} be gateway processes. First, a process p_{11} in G_1 sends a source local message a to all the processes in G_1. Here, $a.seq = 1$, $a.rsq = \langle 1, 2, 1\rangle$, and $a.RSQ = [2, 1]$. The local message a is sent to the gateway process p_{10}. The gateway process p_{10} creates a global message A from a. Here, $A.GSQ = 1$ and $A.RSQ = [2, 1]$. The gateway process p_{10} sends A with $A.RSQ = [2, 1]$ to all the gateway processes in the main subgroup.

RSQ in a gateway process p_{20} is changed to [2, 1]. p_{20} sends a destination local message a_2 of A to all the processes in the subgroup G_2. On receipt of the destination local message a_2, RSQ is changed to [2, 2] in a pair of local processes p_{21} and p_{22} of G_2.

The local process p_{21} sends a source local message b with $b.seq = 1$, $b.rsq = \langle 1, 2, 1\rangle$, and $b.RSQ = [1, 2]$ before receiving the destination local message a_2. The gateway process p_{20} sends a global message B created from b after receiving A. According to the traditional definition, $A \rightarrow B$ since p_{20} sends B after receiving A. However, since the local message b is sent before a_2 is received by p_{21}, a pair of global messages A and B must be causally concurrent. $A.RSQ = [2, 1]$ while $B.RSQ = [1, 2]$. $a \parallel b_1$. $a.rsq = \langle 1, 2, 1\rangle$ and $a.RSQ = [2, 1]$ while $b.rsq = \langle 2, 2, 1\rangle$ and $b.RSQ = [1, 2]$. According to the ordering rules, neither A and B nor a and b_1 are ordered. From Theorem 4, a global message C precedes D even if local messages c and d are causally concurrent in G_2 because c and d are sent in a same subgroup. □

In each subgroup G_i, the vectors of message sequence numbers are used to causally order messages and detect message loss. First, a local process p_{is} sends a message m in G_i [Figure 4]. After receipt of m, another local process sends a message with confirmation of m. A gateway process p_{i0} forwards a global message $M(= g(m))$ to other gateway processes. On receipt of M, a gateway process p_{j0} sends a local message $m_j(= dl_j(M))$. On receipt of m_j, every local process p_{jt} sends a message with confirmation of m_j. If m_j is pre-acknowledged in p_{j0}, p_{j0} sends a global message N with confirmation of M. If M is pre-acknowledged in p_{k0}, p_{k0} sends a local message n_k with confirmation of m. On receipt of the local message n_k, m is pre-acknowledged in every process of G_k. In each local process, messages are ordered according to the ordering rule 1 by the vectors rsq and RSQ as discussed in the preceding section. If a process loses a message m in a subgroup, one process which accepts a message m forwards m to a process which fails to receive m.

5 Evaluation

There are following parameters to evaluate the protocols:

n = number of processes in a group G.

k = number of subgroups.

l_i = number of local processes in each subgroup G_i.

δ_F = delay time in a flat group.

δ_T = delay time in a two-layered group.

In the TG protocol, the size of RSQ is k ($< n$) and the size of rsq is l_i ($< n$) in a subgroup G_i. The overhead of each local process in a subgroup G_i is $O((l_i + k)l_i)$. The overhead for communication among gateway processes is $O(k^2)$ for number k of subgroups. The overhead of a gateway process in a subgroup G_i is $O((l_i + k)l_i + k^2)$.

It takes three rounds to deliver messages in the two-layered group while it takes one round in the flat group. The delay time δ_T in the two-layered group is compared with δ_F in the flat group. In the evaluation, the delay time means duration from time when a process creates a message until time when all the processes receive and process the message in a group.

In a flat group, we consider a pair of processes which are run on a same processor [Figure 5]. In a two-layered group (TG) composed of k subgroups $G_1, \ldots, G_k$, we consider four processes which are run

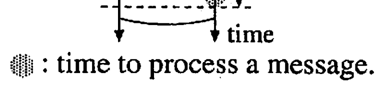

Figure 5: Delay time in flat group.

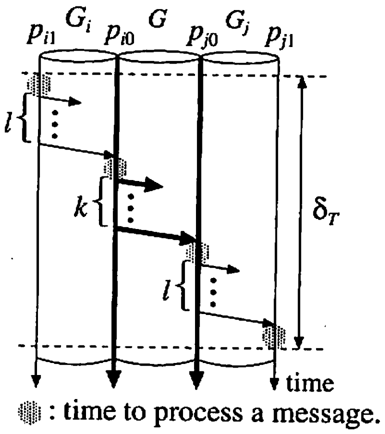

Figure 6: Delay time in two-layered group.

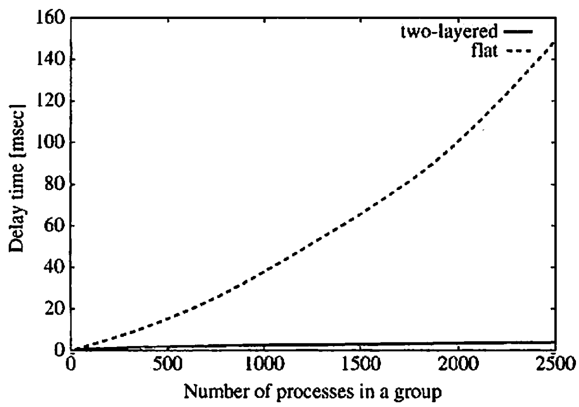

Figure 7: Delay time in one-to-one network.

on a same processor [Figure6]. Here, we assume that every subgroup includes same number l of local processes and $k = \sqrt{n}$. The minimum overhead of a gateway process is obtained for $k = \sqrt{n}$.

First, suppose a process sends a message to each destination process. That is, a process sends n messages in a flat group. A local process sends l local messages and a gateway process sends k global messages in a two-layered group. Figure 7 shows the delay time for number n of processes in a group. The two-layered group implies shorter delay time than the flat group.

Next, suppose a process broadcasts a message in each subgroup. That is, each local process delivers each message to all the local processes including a gateway by one transmission. Figure 8 shows the delay time for number n of processes in a group. If $n \geq 900$, the two-layered group implies shorter delay time than the flat group.

6 Concluding Remarks

We discussed the two-layered group (TG) protocol for large-scale group of processes. In the TG protocol, each message carries a vector whose size is smaller than the total number n of processes. We evaluated the TG protocol in terms of delay time compared with traditional flat group. We showed that the TG protocol implies shorter delay time than the flat group.

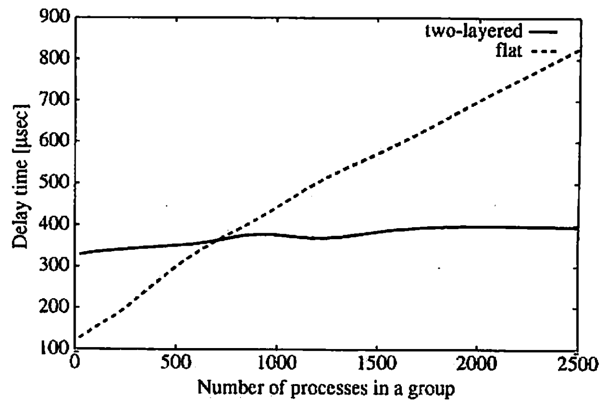

Figure 8: Delay time in broadcast network.

References

[1] K. Birman. Lightweight causal and atomic group multicast. *ACM Trans. on Computer Systems*, pages 272–290, 1991.

[2] S. Deering. Host groups: A multicast extension to the internet protocol. *RFC 966*, 1985.

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

[4] M. F. Kaashoek and A. S. Tanenbaum. An evaluation of the amoeba group communication system. *Proc. of IEEE ICDCS-16*, pages 436–447, 1996.

[5] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *CACM*, 21(7):558–565, 1978.

[6] F. Mattern. Virtual time and global states of distributed systems. *Parallel and Distributed Algorithms*, pages 215–226, 1989.

[7] A. Nakamura and M. Takizawa. Reliable broadcast protocol for selectively ordering pdus. *Proc. of IEEE ICDCS-11*, pages 239–246, 1991.

[8] A. Nakamura and M. Takizawa. Causally ordering broadcast protocol. *Proc. of IEEE ICDCS-14*, pages 48–55, 1994.

[9] T. Tachikawa, H. Higaki, and M. Takizawa. Group communication protocol for realtime applications. *Proc. of IEEE ICDCS-18*, pages 40–47, 1998..

[10] T. Tachikawa, H. Higaki, and M. Takizawa. Δ-causality and ε-delivery for wide-area group communications. *Computer Communications Journal*, 23(1):13–21, 2000.

[11] M. Takizawa, M. Takamura, and A. Nakamura. Group communication protocol for large group. *Proc. of the 18th IEEE Conf. on Local Computer Networks (LCN)*, pages 310–319, 1993.