Regular Paper

Group Communication Protocol for Hierarchical Group

Kojiro Taguchi† and Makoto Takizawa†

A group including a large number of processes implies large computation and communication overheads $O(n^2)$ to manipulate and transmit messages for number n of processes in a group. In this paper, we discuss a hierarchical group (HG) where subgroups of processes are interconnected in order to reduce the overheads. We propose a hierarchical group (HG) protocol to causally deliver messages to processes in the hierarchical group. In the HG protocol, each message carries a vector whose size is the number of subgroups, smaller than number of processes in a group.

1. Introduction

In distributed applications like teleconferences, a collection of multiple processes is cooperating to achieve some objectives. The collection of processes is referred to as *group* [1),5),7),9)~14)]. 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 hundreds of numbers processes. A *wide-area* group is a group where processes are distributed in wide-area networks like the Internet. Tachikawa and Takizawa [12),13)] 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),7)]. In order to support the ordered delivery of messages, a *vector clock* [1),7)] including n elements is used. A header length of a message is $O(n)$ for number n of processes in a group because the message carries the vector clock. Computation and communication overheads are $O(n^2)$ because a process sends a message to all the processes in a group. Even if a group of tens of processes can be realized by traditional group protocols, it is difficult, maybe impossible to support a group of hundreds processes due to large computation and communication overheads. In order to reduce the overheads, *hierarchical* groups are discussed [4),14)]. Papers [2),4)] discuss how to multicast messages in a hierarchical group but do not discuss ordered delivery of messages. Takamura and Takizawa [14)] 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, processes in different local areas establish a subgroup which supports the causally ordered delivery of messages by its own mechanism like physical clock [8)], liner clock [6)], vector clock [1),7)], and centralized controller [5)]. Subgroups are interconnected by the Internet to make a group. We discuss a new type of *hierarchical group* (*HG*) protocol for a large-scale and wide-area group of processes, where each message carries a vector whose size is the number of subgroups, smaller than the total number of processes.

In section 2, we present a system model. In section 3, we discuss the causally ordered delivery of messages in a hierarchical group. In section 4, we discuss the HG protocol. In section 5, we evaluate the HG protocol in terms of computation and communication overheads compared with traditional protocols.

2. System Model

2.1 System Configuration

We present a system configuration of this paper. A system is composed of multiple processes interconnected in 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 [2)], 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, a process sends a message to multiple processes while receiving messages from multiple processes in a

† Department of Computers and Systems Engineering, Tokyo Denki University

group. The membership of the group may be dynamically changed by members' leaving and new members' joining the group [11]. In addition, messages are required to be *causally* delivered to destination processes in the group [1]. Let $s_i(m)$ and $r_j(m)$ denote sending and receipt events of a message m in processes p_i and p_j, respectively. By using the *happens-before* relation [6], the causally precedent relation "$\rightarrow$" on messages is defined: a message m_1 *causally precedes* another message m_2 ($m_1 \rightarrow m_2$) iff $s_i(m_1)$ *happens before* $s_j(m_2)$. A process is required to deliver a message m_1 before another message m_2 if m_1 causally precedes m_2.

Processes are interconnected in networks. Every pair of processes can communicate with one another through a logical communication channel supported by the network. For example, each channel is realized in a connection supported by TCP/IP [3].

2.2 Functions of Group Protocols

It is significant to discuss which process coordinates communication among processes in a group. One way to coordinate communication is a *centralized* way [4),5] 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. The delivery order of messages is decided by the controller. 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, e.g. by using the vector clock [7]. ISIS [1] takes a *decentralized* way where every destination process sends a receipt confirmation to the sender of a message assuming the underlying network is reliable. Takizawa, *et al.* [9),10),13] take a *fully distributed* approach where every destination process sends a receipt confirmation to not only the sender but also all the other destinations by taking usage of less-reliable networks (**Fig. 1**). A process can also detect message loss on receipt of messages including receipt confirmation from other destinations. In order to reduce the 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*. The process sends receipt confirmation of messages 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 (**Fig. 2**), where some destination forwards the message to the process on behalf of the sender [12]. In the other protocols, only the sender retransmits the message.

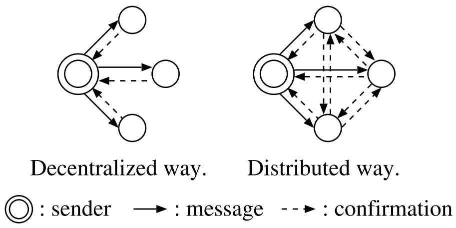

Fig. 1 Transmission of receipt confirmation.

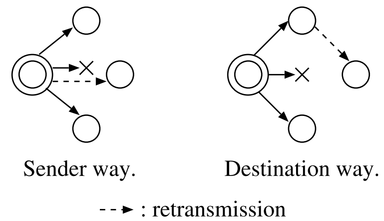

Fig. 2 Retransmission of message.

In traditional distributed group protocols, the vector clock [7] is used in order to causally deliver messages to destination processes in a group. For a group G of n (> 1) processes $p_1, \ldots, p_n$, a vector V is in a form $\langle V_1, \ldots, V_n \rangle$. 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, the ith element V_i is incremented by one, i.e., $V_i := V_i + 1$. Then, the message m carries the vector V ($m.V = \langle m.V_1, \ldots, m.V_n \rangle$) of the sender process p_i. On receipt of a message m from another process, the vector V in a process p_i is manipulated as follows: $V_k := \max(V_k, m.V_k)$ ($k = 1, \ldots, n$, $k \neq i$). Here, a vector $A = \langle A_1, \ldots, A_n \rangle$ is larger than another vector $B = \langle B_1, \ldots, B_n \rangle$ ($A > B$) iff $A_j \geq B_j$ ($j = 1, \ldots, n$) and $A_k > B_k$ for some k. $A \geq B$ iff $A > B$ or $A = B$. A message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$) iff $m_1.V < m_2.V$. m_1 is *causally concurrent* with m_2 ($m_1 \parallel m_2$) iff neither $m_1 \rightarrow m_2$ nor $m_2 \rightarrow m_1$.

2.3 Hierarchical Group

Since the header length of messages is $O(n)$ and the computation and communication overheads are $O(n^2)$, it is difficult, or maybe impossible for the protocol using the vector clock

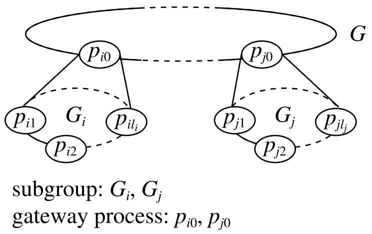

Fig. 3 Model of hierarchical group.

to support a larger group from the performance point of view. One approach to reducing the overheads is to hierarchically construct a group [4),14)]. For example, a group G composed of one hundred processes $p_1, \ldots, p_{100}$ is decomposed into ten subgroups $G_1, \ldots, G_{10}$, each of which includes ten *local* processes. Each subgroup G_i has one process named a *gateway* process p_{i0} ($i = 1, \ldots, 10$). If a process p_{is} in a subgroup G_i sends a message m to destination processes in another subgroup G_j ($j \neq i$), the process p_{is} first sends the message m to a gateway process p_{i0} in G_i. Then, p_{i0} forwards the message m to a gateway process p_{j0} of the destination subgroup G_j. The gateway process p_{j0} delivers the message m to destination processes in G_j. A group G is *hierarchical* iff G is composed of disjoint subgroups and every process in a subgroup does not directly deliver messages to any process in another subgroup (**Fig. 3**). G is *flat* iff G is not hierarchical.

3. Causally Ordered Delivery

We discuss a causality of messages in a hierarchical group. A group G is composed of multiple subgroups $G_1, \ldots, G_k$ ($k > 1$). Each subgroup G_i includes processes $p_{i1}, \ldots, p_{il_i}$ ($l_i \geq 1$) and one gateway process p_{i0}. Processes and messages transmitted in a subgroup are referred to as *local* ones. A subgroup of gateway processes $p_{10}, \ldots, p_{k0}$ is a *main subgroup*. Messages exchanged in the main subgroup are referred to as *global messages*. Suppose a gateway process receives a local message m. A local message which is destined to a process in another subgroup is an *outgoing* one. A global message is created from an outgoing local message by a gateway process. 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 referred to as *source* and *destination* local messages of a global message M, $sl(M)$ and $dl_j(M)$, respectively. A capital letter 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 a subgroup 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 another message M_2 in a main subgroup of G. In each subgroup G_i, local messages can be assumed to be causally ordered by its ordering mechanism. A notation "$m_1 \rightarrow_i m_2$" indicates that a local message m_1 causally precedes another local message m_2 in a subgroup G_i. We discuss how causalities of local messages "$m_1 \rightarrow_i m_2$" and global messages "$g(m_1) \rightarrow_G g(m_2)$" are related.

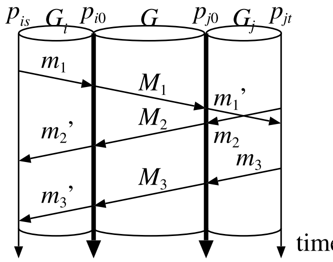

Fig. 4 Causal delivery in hierarchical group.

[Definition] A local message m_1 *causally precedes* another local message m_2 ($m_1 \rightarrow m_2$) iff one of the following conditions holds:

(1) $sl(m_1) \rightarrow_i sl(m_2)$.
(2) $dl_i(m_1) \rightarrow_i sl(m_2)$.
(3) $m_1 \rightarrow m_3 \rightarrow m_2$ for some local message m_3. □

It is straightforward for the following theorem to hold from the definition.

[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. 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 local message m_1 to a process p_{jt} in G_j. A 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 **Fig. 4**. That is, M_1 causally precedes M_2 ($M_1 \rightarrow_G M_2$) but m_1 and m_2 are causally concurrent ($m_1 \parallel m_2$). 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 subgroup of G. "$M_1 \rightarrow_G M_2$" 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 causally ordering a pair of global messages $M_1(= g(m_1))$ and $M_2(= g(m_2))$ in a main subgroup of G unless "$m_1 \rightarrow m_2$" holds.

4. HG Protocol

4.1 Data Transmission

We discuss a basic data transmission procedure of the hierarchical group (HG) protocol for a hierarchical group G composed of multiple subgroups $G_1, ..., G_k$ $(k > 1)$. First, we assume that each subgroup supports some mechanism to causally deliver messages like vector clock.

A local message m exchanged among processes in a subgroup G_i includes following information (**Fig. 5**):

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

A global message M exchanged among gateway processes includes following information (**Fig. 6**):

$M.SG$ = sender subgroup.
$M.DG$ = set of destination subgroups.
$M.VC$ = vector clock $[VC_1, \ldots, VC_k]$.
$M.DATA$ = data (= local message).

Each gateway process p_{i0} is not only a local process in a subgroup G_i but also exchanges global messages with other gateway processes. The gateway process p_{i0} manipulates a *global* sequence number *gseq*. The global sequence number *gseq* shows a sequence number of a global message. A vector $vc = \langle vc_1, \ldots, vc_k \rangle$ manipulated by each local process p_{ij} in G_i is referred to as *local* vector $(j = 0, 1, \ldots, l_i)$. The global sequence number *gseq* and each element in the local vector vc are initially 0 in every process. It is noted that the vector size is the number k of subgroups $(k < n)$.

First, suppose a local process p_{is} in a subgroup G_i sends a local message m to a process p_{jt} in another subgroup G_j. Here, $m.sp = p_{is}$, $m.SG = G_i$, $p_{jt} \in m.dp$, and $G_j \in m.DG$. The process p_{is} sends a source local message m to a gateway process p_{i0} where $m.vc := vc$. It is noted that the local vector vc of the process p_{is} is not updated on sending a local message while the traditional vector clock is incremented on sending a message.

Then, the gateway process p_{i0} receives the outgoing local message m from the process p_{is} in the subgroup G_i. The global sequence number *gseq* in p_{i0} is incremented by one; $gseq := gseq + 1$. Then, a global message $M(= g(m))$ is created from the local message m where $M.VC_i := gseq$, $M.VC_h := m.vc_h$ $(h = 1, \ldots, k,\ h \neq i)$, $M.SG := m.SG$, $M.DG := m.DG$, and $M.DATA := m$. The gateway process p_{i0} sends the global message M to a gateway p_{j0} in each destination subgroup $G_j \in M.DG$.

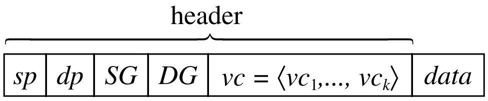

Fig. 5 Local message format.

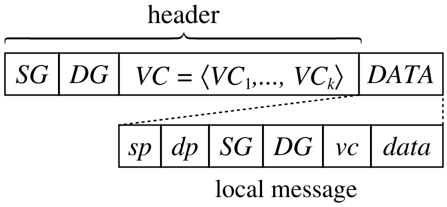

Fig. 6 Global message format.

Next, a gateway process p_{j0} in a subgroup G_j receives a global message M from another subgroup G_i. Here, $vc_h := \max(vc_h,\ M.VC_h)$ $(h = 1, \ldots, k,\ h \neq j)$ in p_{j0}. The gateway process p_{j0} creates a destination local message $m_j(= dl_j(M))$ from the global message M and then forwards m_j to destination processes in G_j. Here, $m_j := M.DATA$ and $m_j.vc := M.VC$. Each gateway process manipulates a pair of local vector vc and global sequence number *gseq* 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} or another local process in a same subgroup G_j. Here, $vc_h := \max(vc_h,\ m.vc_h)$ $(h = 1, \ldots, k,\ h \neq j)$ in p_{jt}.

[Example] **Figure 7** shows a group G composed of three subgroups G_1, G_2, and G_3. Let p_{10}, p_{20}, and p_{30} be gateway processes of the subgroups G_1, G_2, and G_3, respectively. Notations [*gseq*] and $\langle vc_1,\ vc_2,\ vc_3 \rangle$ indicate instances of global sequence number and local vector, respectively, in each process. Initially, $gseq = 0$ and $vc_1 = vc_2 = vc_3 = 0$. First, a process p_{1s} in the subgroup G_1 sends a source 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}. The gateway process p_{10} creates a global message A from the local

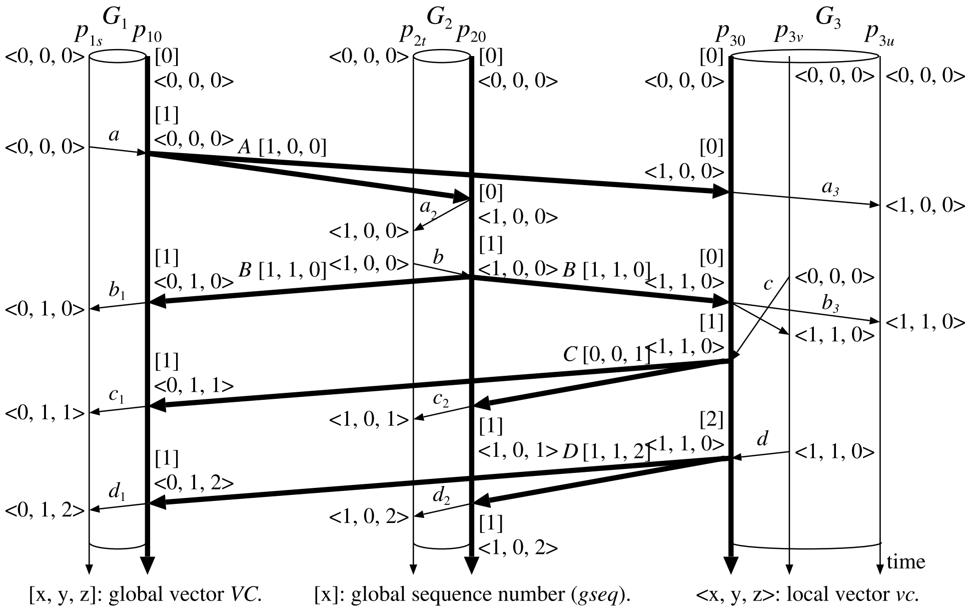

Fig. 7 Communication among subgroups G_1, G_2, and G_3.

message a. Here, $gseq$ of p_{10} is incremented by one and $VC = [1,\ 0,\ 0]$. The gateway process p_{10} sends the global message A with $A.VC = [1,\ 0,\ 0]$ to a pair of gateway processes p_{20} and p_{30}.

The local vectors vc in the gateway processes p_{20} and p_{30} are changed to $\langle 1,\ 0,\ 0\rangle$. The gateway process p_{20} sends a destination 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, the process p_{2t} sends a source local message b with $vc = \langle 1,\ 0,\ 0\rangle$ to the gateway process p_{20}. The global sequence number $gseq$ of p_{20} is incremented by one. The gateway process p_{20} creates a global message B and then sends B to p_{10} and p_{30}. Here, $B.VC = [1,\ 1,\ 0]$. The gateway process p_{10} forwards a destination local message b_1 of a global message B for the local message b with $b.vc = \langle 1,\ 1,\ 0\rangle$. Here, since $a.vc < b_1.vc$, the local message a causally precedes the local message b.

In the subgroup G_3, a process p_{3v} sends a source local message c with $c.vc = \langle 0,\ 0,\ 0\rangle$ before receiving a destination 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, the global message B causally precedes the global message C since the gateway process p_{30} sends C after receiving B. However, since the local message 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 destination local message c_1 of C where $c_1.vc = \langle 0,\ 0,\ 1\rangle$. The destination local message b_1 of B carries the local vector $b_1.vc = \langle 1,\ 1,\ 0\rangle$. Here, the local vectors $\langle 1,\ 1,\ 0\rangle$ and $\langle 0,\ 0,\ 1\rangle$ are not comparable. Here, the local messages b_1 and c_1 are causally concurrent in the process p_{1s}. □

4.2 Ordering of Messages

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

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

[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.

[Proof] Suppose $m_1 \rightarrow m_2$ but $m_1 \not\Rightarrow_i m_2$. If $m_1 \rightarrow m_2$, $g(m_1) \rightarrow_G g(m_2)$ according to Theorem 1. If $g(m_1) \rightarrow_G g(m_2)$, $m_1 \Rightarrow_i m_2$. It contradicts the assumption. □

Even if a global message M_1 causally precedes another global message M_2 in a main subgroup ($M_1 \rightarrow_G M_2$), the causality "$m_1 \rightarrow m_2$" 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. Each subgroup G_i is assumed to support some mechanism like vector clock to causally order local messages. The gateway process 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 HG protocol presented here, the global sequence number $gseq$ of p_{i0} is incremented by one each time the gateway process p_{i0} sends a global message. If p_{i0} sends M_1 before M_2, $dl_j(M_1).vc < dl_j(M_2).vc$ for every common destination subgroup G_j of M_1 and M_2, i.e. $dl_j(M_1)$ precedes $dl_j(M_2)$. Thus, for a pair of local messages m_1 and m_2 sent in a same subgroup, m_1 may precede m_2 even if m_1 and m_2 are causally concurrent. Thus, the following theorem holds.

[Theorem 3] A local message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$) if m_1 precedes m_2 in a subgroup G_i ($m_1 \Rightarrow_i m_2$) and $m_1.SG \neq m_2.SG$, i.e. m_1 and m_2 are sent in different subgroups. □

5. Evaluation

In traditional protocols, computation and communication overheads are $O(n^2)$ for number n of processes in a group. Because a process causally order messages by using the vector clock and sends a message to all the processes in the flat group. In the HG protocol, the overhead for communication among gateway processes is $O(k^2)$ for number k of subgroups ($k < n$). The overhead of each subgroup G_i is $O(l_i^2)$ for number l_i of processes in a subgroup G_i ($l_i < n$).

It takes three rounds to deliver messages in the hierarchical group while it takes one round in the flat group. The round trip time RTT_H in the HG protocol is compared with RTT_F in the flat group protocol. The round trip time is obtained by summing message delay time in the networks and processing time in processes which a message passes over. In the evaluation, the round trip time of each message is duration from time when a process sends a message until time when the process receives a response message from the destination process (**Fig. 8**). There are following parameters to evaluate the protocols:

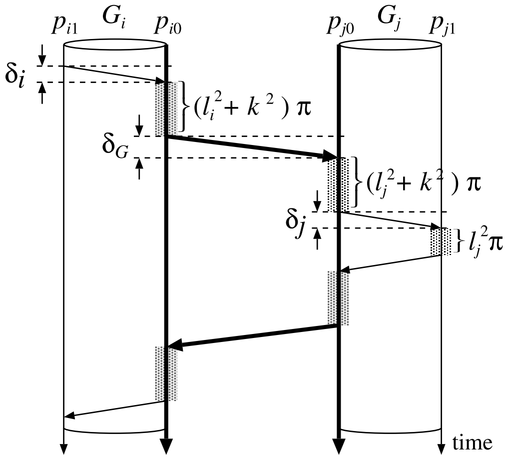

Fig. 8 Round trip time.

- n = number of processes $p_1, \ldots, p_n$ in a group G.
- k = number of subgroups $G_1, \ldots, G_k$.
- l_i = number of processes in a subgroup G_i.
- δ_{ij} = delay time between a pair of processes p_i and p_j in a flat group.
- δ_i = delay time between every pair of processes in a subgroup G_i, assuming $\delta_{st} = \delta_i$ for every pair of local process p_{is} and p_{it} in G_i.
- δ_G = delay time between a pair of gateway processes p_{i0} and p_{j0} in a main subgroup.
- π = time units to process one unit work to handle a message, e.g. time to process one element in a vector.

In a flat group, it takes $n^2\pi$ time units to send a message after receiving another message. Hence, the round trip time RTT_F in a flat group is given as follows:

$$RTT_F = n^2\pi + 2\delta_{ij}. \quad (1)$$

Next, let us consider a hierarchical group composed of k subgroups $G_1, \ldots, G_k$. Here, a process p_{is} sends a local message m to a gateway process p_{i0}. Secondly, the global message is forwarded to a destination gateway process p_{j0}. Then, the gateway process p_{j0} forwards the local message to a destination process p_{jt}. The round trip time RTT_H is given as follows (Fig. 8):

$$RTT_H = 2\ \left(\delta_i + \left(l_i^2 + k^2\right)\pi + \delta_G + \left(l_j^2 + k^2\right)\pi + \delta_j\right) + l_j^2\pi. \quad (2)$$

Here, we assume that every subgroup G_i includes same number of processes, $l_i = l$ and delay time between every pair of processes in G_i is same, $\delta_i = \delta$. RTT_H is given as follows:

$$RTT_H = \left(5l^2 + 4k^2\right)\pi + 4\delta + 2\delta_G. \quad (3)$$

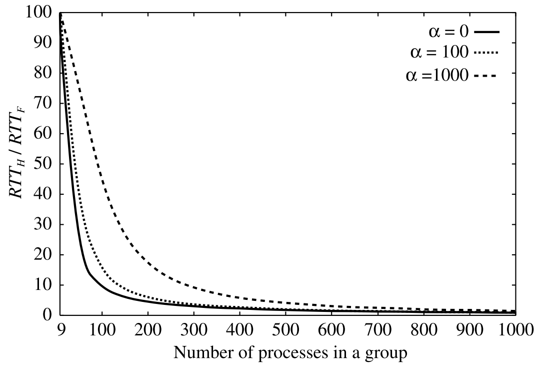

Fig. 9 RTT ratio ($\alpha_G = \alpha$).

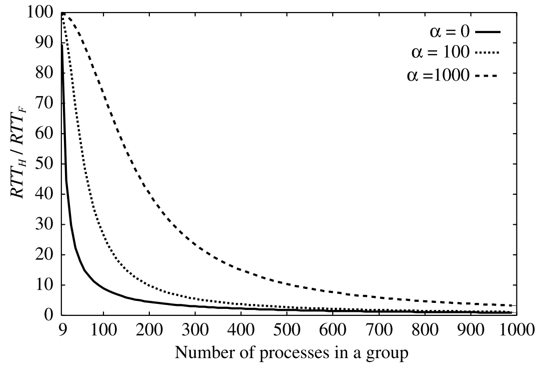

Fig. 10 RTT ratio ($\alpha_G = 10\alpha$).

Since $n = kl$, the following formula is derived.

$$RTT_H = \left(\frac{5n^2}{k^2} + 4k^2\right)\pi + 4\delta + 2\delta_G. \quad (4)$$

The minimum value of RTT_H is given for $k = (5/4)^{-1/4}\sqrt{n}$.

$$RTT_H = 4\sqrt{5}n\pi + 4\delta + 2\delta_G. \quad (5)$$

If a flat group is realized by a same network topology as the hierarchical group, $\delta_{ij} = 2\delta + \delta_G$, for every pair of processes p_i and p_j. Let $\delta = \alpha\pi$ and $\delta_G = \alpha_G\pi$ for some constants α and α_G. α and α_G show ratios of communication speed to processing speed.

$$\begin{aligned} RTT_F &= n^2\pi + 4\delta + 2\delta_G \\ &= \left(n^2 + 4\alpha + 2\alpha_G\right)\pi. \quad (6) \end{aligned}$$

$$RTT_H = \left(4\sqrt{5}n + 4\alpha + 2\alpha_G\right)\pi. \quad (7)$$

First, we discuss a case subgroups and main subgroup take usage of a same type of network, i.e. $\alpha = \alpha_G$. **Figure 9** shows a ratio of RTT_H to RTT_F for $\alpha = 0$, 100, 1000. $\alpha = 1000$, $\alpha = 100$, and $\alpha = 0$ show three types of networks, slower to faster ones (Fig. 9). If $n \geq 9$, the hierarchical group implies shorter round trip time than the flat group. For example, in case $n = 100$, the round trip time is reduced to 9 [%] for $\alpha = 0$, 14 [%] for $\alpha = 100$, and 43 [%] for $\alpha = 1000$.

Next, let us consider a hierarchical group where local processes are interconnected in each subgroup with local area networks and subgroups are interconnected with the Internet. Here, $\alpha_G = 10\alpha$ (**Fig. 10**). For example, in case $n = 100$, the round trip time is reduced to 9 [%] for $\alpha = 0$, 27 [%] for $\alpha = 100$, and 73 [%] for $\alpha = 1000$.

6. Concluding Remarks

We discussed the group protocol named *HG* protocol for a large-scale group of processes. A group is hierarchically structured in a family of subgroups of processes which are interconnected. In the HG protocol, each message carries a vector of k elements for number k of subgroups which is smaller than the total number n of processes. We evaluated the HG protocol in terms of message header length and response time compared with traditional flat group. We showed that the HG protocol implies shorter response time than the flat group.

References

1) Birman, K.: Lightweight Causal and Atomic Group Multicast, *ACM Trans. Comput. Syst.*, pp.272–290 (1991).
2) Deering, S.: Host Groups: A Multicast Extension to the Internet Protocol, RFC 966 (1985).
3) Defense Communications Agency: *DDN Protocol Handbook*, Vol.1-3, NIC 50004-50005 (1985).
4) Hofmann, M., Braun, T. and Carle, G.: Multicast Communication in Large Scale Networks, *Proc. IEEE HPCS-3* (1995).
5) Kaashoek, M.F. and Tanenbaum, A.S.: An Evaluation of the Amoeba Group Communication System, *Proc. IEEE ICDCS-16*, pp.436–447 (1996).
6) Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System, *CACM*, Vol.21, No.7, pp.558–565 (1978).
7) Mattern, F.: Virtual Time and Global States of Distributed Systems, *Parallel and Distributed Algorithms*, pp.215–226 (1989).
8) Mills, D.L.: Network Time Protocol, RFC 1305 (1992).
9) Nakamura, A. and Takizawa, M.: Reliable Broadcast Protocol for Selectively Ordering

PDUs, *Proc. IEEE ICDCS-11*, pp.239–246 (1991).
10) Nakamura, A. and Takizawa, M.: Causally Ordering Broadcast Protocol, *Proc.IEEE ICDCS-14*, pp.48–55 (1994).
11) Reiter, M.K.: The Rampart Toolkit for Building High-Integrity Services, *Theory and Practice in Distributed Systems*, LNCS 938, pp.99–110, Springer-Verlag (1995).
12) Tachikawa, T., Higaki, H. and Takizawa, M.: Group Communication Protocol for Realtime Applications, *Proc. IEEE ICDCS-18*, pp.40–47 (1998).
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, pp.13–21 (2000).
14) Takizawa, M., Takamura, M. and Nakamura, A.: Group Communication Protocol for Large Group, *Proc. 18th IEEE Conf. on Local Computer Networks (LCN)*, pp.310–319 (1993).

(Received May 27, 2002)
(Accepted December 3, 2002)

Kojiro Taguchi was born in 1979. He received his B.E. degree in Computers and Systems Engineering from Tokyo Denki University, Japan in 2001. He is now a graduate student of Tokyo Denki University. His research interests include group communication protocols and distributed systems.

Makoto Takizawa is a full professor in the Department of Computers and Systems Engineering, Tokyo Denki University, Japan. He is now a dean of the graduate school of Science and Engineering, Tokyo Denki University. He chaired the Information Division at the Research Institute for Technology, Tokyo Denki University from 1998 to 2002. He was a visiting professor at GMD-IPSI, Germany (1989–1990) and has been a regular visiting professor at Keele University, England since 1990. He is a fellow of Information Processing Society of Japan (IPSJ) and was a member of the executive board of IPSJ from 1998 to 2000. He chaired SIGDPS (distributed processing) of IPSJ from 1997 to 2000 and was an editor of the Journals of IPSJ (1994–1998). He received his BE and ME in applied physics, and DE in computer science from Tohoku University, Japan. In 1996, he won the best paper award at IEEE International Conference on Parallel and Distributed Systems (ICPADS). He was a general co-chair of IEEE ICDCS-2002 and a program co-chair of IEEE ICDCS-1998. He is a founder of ICOIN and AINA conferences. He was elected for 2003–2005 BoG member of IEEE Computer Society. He is a member of the IEEE and a member of the ACM and IPSJ. His research interests include distributed systems, group communication protocols, distributed objects, fault-tolerant systems, and information security.