Regular Paper

Group Communication Protocol for Wide-area Groups

TAKAYUKI TACHIKAWA† and MAKOTO TAKIZAWA†

Group communication protocols support the ordered and reliable delivery of messages to multiple destinations in a group of processes. The group communication protocols discussed hitherto assume that the delay time between every two processes is almost the same. In world wide applications using the Internet, it is essential to consider a wide-area group in which the delay times among the processes are significantly different. After defining a Δ^*-causality to be applied in a wide-area group, we present a protocol that supports a group of processes with the Δ^*-causality, and evaluate it in the world wide environment.

1. Introduction

Distributed systems are composed of multiple computers interconnected by communication networks. In distributed applications such as teleconferencing, a group of multiple processes, that is, a process group [18], is established and the processes in the group are executed cooperatively. Group communication protocols support a group of processes with reliable and ordered delivery of messages to multiple destinations. Transis [3], ISIS (CBCAST) [6], Psync [22], and other protocols [2),21),29] support causally ordered delivery, while Totem [4], ISIS (ABCAST) [6], Ameoba [16], Trans/Total [20], Rampart [26], and others [7),27] support totally ordered delivery. Some systems [9),10] support both.

The group communication protocols discussed hitherto assume that all pairs of processes have almost the same delay time and reliability. Here, let us consider a world wide teleconference among five processes K, U, O, T, and H at Keele in UK, UCLA and Ohio in the USA, and Tokyo and Hatoyama in Japan, respectively. On the Internet, it takes about 60 msec to propagate a message within Japan, while it takes about 240 msec to do so between Tokyo and Europe. The longer the distance, the more messages are lost. For example, about 20% of the messages are lost between Japan and Europe while fewer than 1% are lost in Japan. Thus, it is essential to consider a group communication where the delay times between the processes are significantly different [12),13),15], that is, non-negligible in relation to the processing speed. Such a group of processes is named a *wide-area* group. In a wide-area group, the time needed to deliver messages to the destinations is determined by the longest delay between the processes. For example, if T sends a message m to H and K, T has to wait for a response from K after having received a response from H. Next, suppose that K sends a message m to H and T, respectively. If T loses m, T requires the sender K to resend it. The delay time between T and K is about four times longer than that between T and H. If H resends m, the time needed to retransmit m can be reduced. Thus, the delivery time by retransmitting a message from another destination, that is, the *destination* retransmission can be reduced.

Suppose that T sends m to H, U, and K. On receiving m, the destination processes send receipt confirmation messages to T. Here, let us consider a method whereby K sends the confirmation to U, instead of directly to T, and then U sends the confirmation back to T. Even if U loses m, the delay time can be reduced if K retransmits m to U as described above. A wide-area group G can be decomposed into disjoint subgroups $G_1, \cdots, G_{sg}$ $(sg \geq 2)$ [12),30] where each G_i includes processes close to each other and has one coordinator process. Messages sent by a process are exchanged by the coordinators of the subgroups. In Holbrook, et al. [13], each subgroup has a log in which transmitted messages are recorded.

In multimedia and real-time applications, messages have to be delivered in some predetermined time units. We will discuss the Δ-causality [1),5),31], where Δ denotes the maximum delay time between the processes. That is, it is meaningless to receive a message m unless m is delivered within a time Δ of being transmitted. The Δ-causality assumes that ev-

† Department of Computers and Systems Engineering, Tokyo Denki University

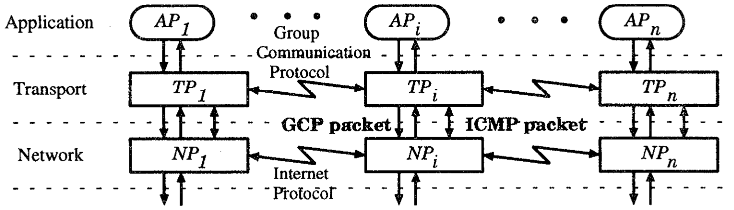


Fig. 1 System model.

ery process has the same Δ. In world wide applications, the maximum delay time Δ_{ij} is specified for each pair of processes P_i and P_j and the difference between some Δ_{ij} and Δ_{kl} is not negligible. In this paper, we define a Δ^*-*causality* where some pair of Δ_{ij} and Δ_{kl} are significantly different. For example, Δ_{HK} is four times longer than Δ_{HT}. Then, we present a protocol supporting the Δ^*-causality, which is realized by the destination and replicated retransmission.

In Section 2, we present a system model. In Section 3, we present protocols in the wide-area group. In Section 4, we discuss the Δ^*-causality. We evaluate the protocols in Section 5.

2. System Model

A distributed system is composed of three hierarchical layers, namely, the *application*, *transport*, and *network* layers, as shown in **Fig. 1**. A group of $n(\geq 2)$ application processes $AP_1, \cdots, AP_n$ are executed cooperatively. Each AP_i communicates with other processes in the group by using the underlying group communication service provided by transport processes $TP_1, \cdots, TP_n$. Here, let G be a group of the transport processes ($G = \{TP_1, \cdots, TP_n\}$). G is considered to support each pair of processes TP_i and TP_j with a logical channel. Data units transmitted at the transport layer are *packets*. TP_i sends a packet to TP_j via the channel. The network layer provides the IP service[24] for the transport layer.

The cooperation of the processes at the transport layer is coordinated by *group communication* (GC) and *group communication management* (GCM) protocols. The GC protocol establishes a group G and reliably and causally[6] delivers packets to the destination processes in G. The GCM protocol is used for monitoring and managing the membership of G. An application process AP_i requests TP_i to send an application message s. TP_i decomposes s into packets, and sends them to multiple destinations in G. The destination process TP_j assembles the packets into a message s_j, and delivers s_j to AP_j. Packets decomposed from the application message are *messages*.

A transport process TP_i has to know the delay time δ_{ij} with each TP_j in G. In the GCM protocol, TP_i requests the network layer to transmit two kinds of ICMP[25] packets: "Timestamp" and "Timestamp Reply." By using the time information, TP_i can know when "Timestamp" sent by TP_i is received by TP_j and when "Timestamp Reply" received by TP_i is sent by TP_j. TP_i calculates δ_{ij}, the round trip time, and periodically sends the ICMP packets to all the processes in G. Here, TP_j is referred to as *nearer* to TP_i than TP_k if $\delta_{ij} < \delta_{ik}$. In addition, the GCM protocol monitors the ratio ε_{ij} of packets lost between each pair of TP_i and TP_j. Here, we assume that $\delta_{ij} = \delta_{ji}$ and $\varepsilon_{ij} = \varepsilon_{ji}$ for every pair of TP_i and TP_j.

We make the following assumptions about packets sent by TP_i:

- Packets may be lost and duplicated.
- Packets can be sent to any subset V of destination processes in a group G ($V \subseteq G$).
- Packets sent to V are not received by processes that are not included in V.
- Packets sent by the same process may be received by the destination processes in an order different from that in which they were sent (not assuming FIFO).

3. Reliable Receipt

3.1 Transmission and Confirmation

In group communication, a message m sent by one process TP_i is sent to multiple destination processes in a group $G = \{TP_1, \cdots, TP_n\}$. m has to be *reliably* delivered to all the destinations in G. Here, let s be the number of destinations of m. Two points need to be resolved for reliable receipt of m:

(1) how to deliver m to its destinations, and

(2) how to deliver confirmation of the receipt of m to the sender TP_i and the destinations.

As regards the first point, there are two delivery schemes: *direct* and *hierarchical*. In direct multicasting, TP_i sends m directly to all the destinations. In hierarchical multicasting, TP_i sends m to a subset of the destinations. On receipt of m from TP_i, TP_j forwards m to other destinations. Propagation-tree-based routing algorithms have been proposed for this purpose [11),14)]. Another approach is to decompose G into disjoint subgroups $G_1, \cdots, G_{sg}$ ($sg \geq 2$) [30)]. Each G_i has one coordinator process. TP_i sends m to the coordinator, and the coordinators forward m to the destinations in the subgroups.

There are two schemes for delivering the confirmation: *decentralized* and *distributed*. In the decentralized scheme [6)], TP_i sends m to the destinations and the destinations send back confirmation of the receipt of m to TP_i. If TP_i receives all the confirmations, TP_i informs all the destinations of the reliable receipt of m. A total of $3s$ messages are transmitted, three rounds are needed.

In the distributed scheme [27),29)], every destination TP_j sends the receipt confirmation of m to all the destinations and TP_i on receipt of m. If each TP_j receives confirmations from all the destinations, TP_j reliably receives m. Here, $O(s^2)$ messages are transmitted, and two rounds are needed. In Tachikawa and Takizawa [29)], the number of messages transmitted in G can be reduced to $O(s)$ by adopting the *piggy back* and the *deferred confirmation* schemes.

The following protocols can be used to realize the receipt confirmation of m:

(1) Direct multicast and distributed confirmation.

(2) Direct multicast and decentralized confirmation.

(3) Hierarchical multicast and distributed confirmation.

(4) Hierarchical multicast and decentralized confirmation.

The first one is named a *distributed* protocol [21)]. The second, named *decentralized* protocol, is used in ISIS [6)] and other protocols [3),16),20)].

Next, we consider when each destination process can deliver the messages received. Here, let m_1 be a message received by TP_i. TP_i can deliver m_1 if (1) TP_i has delivered every message m_2 such that $m_2 \rightarrow m_1$ and (2) m_1 is reliably received by all the destinations. How long it takes to reliably receive messages depends on the maximum delay time among the processes in G. Hence, the delay in delivering messages is increased if G includes more distant processes. Since the processes are assumed not to be faulty, messages are eventually reliably received by all the destinations. Hence, TP_i can deliver m_1 if TP_i delivers every message m_2 destined to TP_i such that $m_2 \rightarrow m_1$ even if m_1 is not reliably received. The reliable receipt of m_1 is required to realize the following points:

(1) A message m is guaranteed to be buffered by at least one process TP_j in G. Hence, if m is lost by some process, it can be retransmitted by TP_j.

(2) m can be removed from the buffer if it is reliably received, that is, there is no need to retransmit m.

Hence, only the sender or destination of a retransmitted m needs to know whether or not m is reliably received.

3.2 Recovery and Prevention of Message Loss

In the underlying network, messages are lost due to buffer overruns, unexpected delay, and congestion. Hence, the processes have to recover from message loss. Let us consider a group $R = \{H, U, O, K\}$. **Figure 2** shows a *process graph* of R in which each node denotes a process and each edge $\langle a, b\rangle$ shows a channel between nodes a and b. The weight of $\langle a, b\rangle$ indicates the average delay time δ_{ab}. In Fig. 2 (2), a directed edge $a \rightarrow b$ means that b is the nearest to a. Suppose that H sends a message m to U, O, and K, but O fails to receive m. In

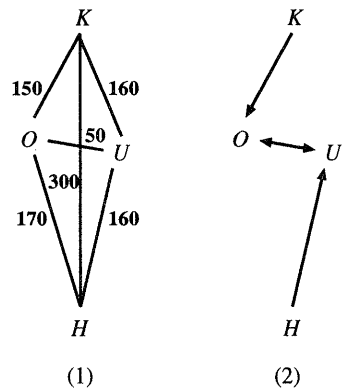


Fig. 2 Process graph of the group H.

traditional protocols, the sender H retransmits m to O and it takes $2\delta_{HO}$. On the other hand, if U forwards m to O, it takes $2\delta_{UO}$. Since $\delta_{HO} > \delta_{UO}$, we can reduce the time needed for retransmission of m if U forwards m to O. Thus, if a process TP_j loses m, the TP_k with the smallest δ_{kj} can send m to TP_j. Thus, there are two ways of retransmitting m if TP_j loses the m sent by TP_i.

(1) Sender retransmission: TP_i retransmits m to TP_j.

(2) Destination retransmission: some destination process TP_k forwards m to TP_j.

In Fig. 2, if H sends m to U more than once, U can receive one replica of m even if U loses some of the replicas. Thus, one way to prevent from the message loss is for the sender TP_i of m to send multiple replicas of m to the destinations. Another way is for a destination TP_j to forward m to another destination TP_k while TP_i sends m to TP_k. TP_k receives m from TP_i and TP_j. For example, U sends m to O on receipt of m, while H directly sends m to O. O can receive m from U even if the m sent by H is lost. The former way is named *direct* replication and the latter *indirect* replication. Protocols with direct or indirect replication are named *replicated* protocols.

3.3 Protocols

Suppose that a process TP_i sends m to a subset V_m of the destination processes in the group G. The following protocols can be used:

(1) Basic (B) protocol: distributed protocol with sender retransmission.

(2) Modified (M) protocol: distributed protocol with destination retransmission.

(3) Nested group (N) protocol: hierarchical multicast and decentralized confirmation with destination retransmission.

(4) Decentralized (D) protocol: direct multicast and decentralized confirmation with sender retransmission.

Each protocol can be replicated or non-replicated.

[Basic (B) protocol]

(T1) TP_i sends m to every destination process in V_m ($\subseteq G$).

(T2) On receipt of m, each process TP_j in V_m sends the receipt confirmation to TP_i.

(T3) On receipt of the confirmation messages from all the processes in V_m, TP_i reliably receives m.

(R) If some TP_j fails to receive m, TP_i sends m to TP_j again. □

The modified (M) protocol is the same as B except that the destination retransmission is adopted.

[Modified (M) protocol]

(R) If TP_j fails to receive m, some destination TP_k nearest to TP_j sends m to TP_j. If all the destinations lose m, T1 is executed again. □

In the N protocol, G is decomposed into disjoint subgroups $G_1, \cdots, G_{sg}$ ($sg \geq 2$). Each G_i is composed of the processes $TP_{i1}, \cdots, TP_{ih_i}$ ($h_i \geq 1$) where TP_{i1} is a coordinator.

[Nested group (N) protocol]

(T1) TP_{ij} sends m to the coordinator TP_{i1}. Let DC_i be the set of coordinators whose subgroups include the destinations of m. TP_{i1} forwards m to the coordinators in DC_i.

(T2) On receipt of m, the coordinator TP_{k1} sends m to the destinations in G_k. On receipt of m, the destination TP_{kh} sends the confirmation back to TP_{k1}. On receipt of the confirmations from all the destinations in G_k, TP_{k1} sends the confirmation to the coordinators in DC_i.

(T3) On receipt of the confirmations from all coordinators in DC_i, TP_{k1} sends the confirmation to the destinations in G_k. On receipt of the confirmation from TP_{k1}, TP_{kh} reliably receives m.

(R) If TP_{kh} fails to receive m, TP_{k1} resends m to TP_{kh}. □

In the D protocol, only the sender TP_i can know whether each destination receives m or not. Hence, sender retransmission is adopted. T1 and R are the same as the B protocol.

[Decentralized (D) protocol]

(T2) On receipt of m, TP_j sends a confirmation back to TP_i.

(T3) On receipt of all the confirmations, TP_i sends an acceptance to all the processes in B.

(T4) On receipt of the acceptance, TP_j accepts m. □

Figures 3 (1), (2), and (4) show the B, M, and D protocols where H sends a message m to U, O, and K but K loses m. In the M protocol, O forwards m to K, since O is the nearest to K. **Figure 3** (3) shows the N protocol with three subgroups $\langle H \rangle$, $\langle U, O \rangle$, and $\langle K \rangle$, where H, U,

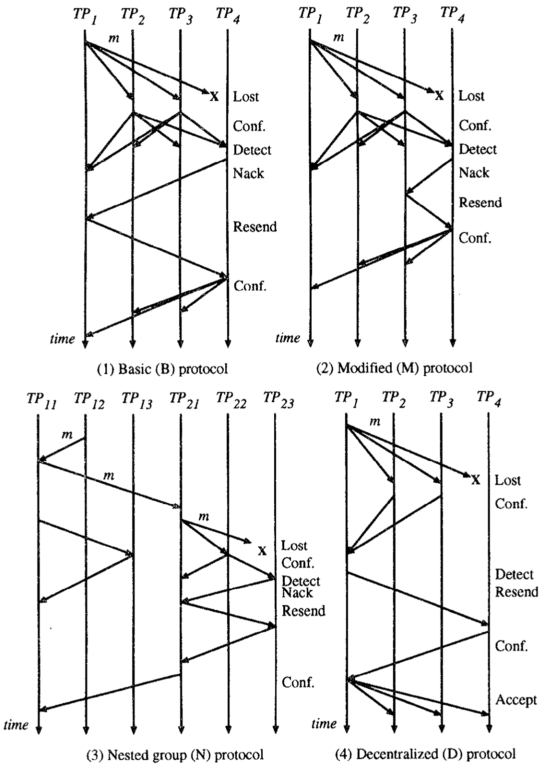


Fig. 3 Protocols.

and O are the coordinators. U receives m but O loses m. Here, U resends m to O.

In the B and D protocols, only H is required to buffer m, since H retransmits m. All the processes may retransmit m. Hence, every process has to buffer m. In the N protocol, H sends m only to U, and then U forwards m to O and K. If either O or K loses m, U retransmits m. Hence, the coordinators have to have buffers. The B and D protocols require fewer buffers than the others.

4. Δ^*-Causality

4.1 Δ-Causality

The messages sent in a group $G=\{TP_1,\cdots,TP_n\}$ have to be delivered in causal order [6].

[Causal precedence relation] A message m_1 and m_2, m_1 *causally precedes* m_2 ($m_1 \rightarrow m_2$) iff

- m_1 is sent before m_2 by a process,
- m_2 is sent after a has been delivered by a process, or
- for some message m_3, $m_1 \rightarrow m_3 \rightarrow m_2$. □

The messages can be causally ordered by using the vector clock [19].

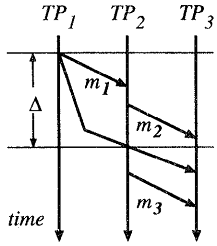


Fig. 4 Δ-causality.

In real-time applications such as multimedia communications, messages have to be delivered to their destinations by a deadline. Thus, TP_j has to receive a message m in Δ time units after TP_i sends m [1),5),31)]. Δ denotes the maximum delay time between the processes in G. Here, let $ts(m)$ be the time when m is sent. Let $tr_i(m)$ be the time when TP_i receives m. Suppose that TP_i sends a message m to TP_j; then m is referred to as *received in* Δ by TP_j iff $ts(m)+\Delta \geq tr_j(m)$. That is, m is received in Δ after m is sent. The causality based on Δ [1)] is defined as follows:

[Δ-causality] For every pair of messages m_1 and m_2, m_1 Δ-*causally precedes* m_2 ($m_1 \overset{\Delta}{\rightarrow} m_2$) iff

(1) $m_1 \rightarrow m_2$ and (2) $ts(m_1)+\Delta \geq ts(m_2)$. □

In a group $K=\{TP_1,TP_2,TP_3\}$ as shown in **Fig. 4**, TP_1 sends a message m_1 to TP_2 and TP_3. TP_2 sends m_2 after receiving m_1 in Δ ($m_1 \overset{\Delta}{\rightarrow} m_2$). Then, TP_2 sends m_3 to TP_3. TP_3 receives m_2 in Δ after m_2 is sent but receives m_1 not in Δ. Hence, TP_3 delivers m_2 but not m_1.

4.2 Δ^*-Causality

In a wide-area group $G=\{TP_1,\cdots,TP_n\}$, some pairs of delay times δ_{ij} and δ_{kh} are significantly different. The application requires that messages sent by TP_i be delivered to TP_j in D_{ij} time units. Here, let Δ_{ij} be obtained on the basis of the statistics of the δ_{ij} between TP_i and TP_j and the requirement D_{ij} of the application. For example, Δ_{ij} may be an average of δ_{ij} if $\Delta_{ij} \leq D_{ij}$. If the distance between TP_i and TP_j is larger than TP_k and TP_l, $\Delta_{ij} \geq \Delta_{kl}$. Let Δ^* be a set $\{\Delta_{ij} \mid i,j=1,\cdots,n\}$.

[Δ^*-causality] Let m_1 and m_2 be messages sent by TP_i and TP_j, respectively. m_1 Δ^*-*causally precedes* m_2 ($m_1 \overset{\Delta^*}{\rightarrow} m_2$) iff

(1) $m_1 \rightarrow m_2$ and (2) $ts(m_1)+\Delta_{ij} \geq$

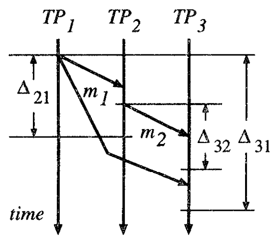


Fig. 5 Δ^*-causality.

ts (m_2). □

That is, m_2 is sent within Δ_{ij} time units of m_1.

In **Fig. 5**, TP_1 sends a message m_1 to TP_2 and TP_3, and TP_2 sends m_2 to TP_3 after receiving m_1. Since TP_3 receives m_2 in Δ_{32}, TP_3 delivers m_2. Then, TP_3 receives m_1. Since TP_3 receives m_1 in Δ_{31}, TP_3 can deliver m_1. However, since m_1 has already been delivered and $m_1 \overset{\Delta^*}{\rightarrow} m_2$, TP_3 cannot deliver m_1. If m_1 is delivered, m_2 cannot be delivered, because m_2 must be delivered after $ts\,(m_2) + \Delta_{32}$. There is inconsistency between Δ_{12} and Δ_{23}. This example shows that TP_i may not deliver m even if m is received in Δ_{ij}. Thus, the Δ^*-causality may be inconsistent if each Δ_{ij} is independently decided.

[**Consistency**] The Δ^*-causal precedence relation $\overset{\Delta^*}{\rightarrow}$ is *consistent* iff for every pair of messages m_1 and m_2 sent by processes TP_i and TP_j, respectively, $ts\,(m_1)+\Delta_{ki} \leq ts\,(m_2)+\Delta_{kj}$ and $m_1 \rightarrow m_2$. □

It can be straightforwardly shown that the following theorem holds:

[**Theorem**] $\overset{\Delta^*}{\rightarrow}$ is consistent if for every triplet of processes TP_i, TP_j, and TP_k, $\Delta_{ij} + \Delta_{jk} \geq \Delta_{ik}$. □

[**Collorary**] $\overset{\Delta^*}{\rightarrow}$ is consistent if for every triplet of processes TP_i, TP_j, and TP_k, $\Delta_{ij} = \Delta_{kj}$. □

That is, the Δ-causality $\overset{\Delta}{\rightarrow}$ is consistent, because $\Delta_{ij} = \Delta$ for every TP_i and TP_j.

In the wide-area group, the theorem may not hold depending on the routing strategies; that is, Δ^* may be inconsistent. The following ways of resolving the inconsistency on the Δ^*-causality exist:

(1) neglecting messages that do not satisfy the Δ^*-causality,

(2) changing some Δ_{ij} so that Δ^* is consistent, and

(3) indirectly replicating messages.

First, let us consider the example of Fig. 5. TP_3 receives m_2 in Δ_{23} and m_1 in Δ_{13}. One way of resolving the inconsistency is for TP_3 to deliver m_2 exactly Δ_{23} after m_2 is sent; that is, m_1 is rejected. The other way is to wait for m_1. As a result, m_2 is rejected, since m_2 is received after $ts\,(m_2) + \Delta_{23}$. In the latter case, neither m_1 nor m_2 is received, although m_2 can be received if m_1 is lost.

For each TP_i, suppose that $\min(\Delta_{1i}, \cdots, \Delta_{ni}) \leq \Delta_i \leq \max(\Delta_{1i}, \cdots, \Delta_{ni})$. TP_i buffers messages received. Let T_i be a variable showing the current time in TP_i. If there is a message m from TP_j in the buffer such that $ts\,(m) + \Delta_i = T_i$ and $ts\,(m) + \Delta_{ji} < T_i$, m is delivered. The smaller Δ_i becomes, the more messages from the more distant processes are rejected. In our implementation, we assume that real-time data are more often exchanged between processes that are nearer to each other. Hence, Δ_i is $\min(\Delta_{1i}, \cdots, \Delta_{ni})$.

Next, we discuss how to obtain a consistent precedence Δ^+ from Δ^* if Δ^* is inconsistent. Δ^*_{ij} is defined to be the minimum delay time among the paths from TP_i to TP_j. If the theorem holds, $\Delta^*_{ij} = \Delta_{ij}$. Otherwise, $\Delta^*_{ij} = \Delta^*_{ik} + \Delta_{kj}$ for some TP_k. We define the following set Δ^+ from Δ^*.

- $\Delta^+ = \{\Delta^+_{ji} \mid \Delta^+_{ji} = \Delta_{ji}$ if $\Delta^*_{ki} + \Delta_{jk} \geq \Delta_{ji}$ for every k; otherwise, $\Delta^+_{ji} = \max(\{\Delta^*_{ki} + \Delta_{jk} \mid \Delta^*_{ki} + \Delta_{jk} \geq \Delta_{ji}$ for every $k\})\}$.

It is clear that $\overset{\Delta^+}{\rightarrow}$ is consistent, because $\Delta^+_{ij} + \Delta^+_{jk} \geq \Delta^+_{ik}$ for every i, j, and k. However, $\Delta^+_{ji} > \Delta_{ji}$ for some TP_i and TP_j. Even if TP_i receives m from TP_j in Δ^+_{ji}, it might be too late to deliver m to the application.

Let us consider another way to transmit redundantly messages so as to satisfy Δ^*. If TP_2 sends m_2 with m_1 to TP_3, TP_3 receives m_1 in Δ_{31} even if the time taken by m_1 sent by TP_1 to arrive at TP_3 is not Δ_{31} as in Fig. 5. In addition, if the channel $\langle TP_1, TP_3\rangle$ is less reliable, TP_3 may lose m_1. Hence, if $\langle TP_1, TP_2\rangle$ and $\langle TP_2, TP_3\rangle$ are more reliable than $\langle TP_1, TP_3\rangle$; that is, if $(1-\varepsilon_{12}) \cdot (1-\varepsilon_{23}) > (1-\varepsilon_{13})$, TP_3 can more reliably receive m if TP_2 forwards m to TP_3. Here, suppose that TP_i sends m to TP_j. Let M_{kj} be a set of messages that TP_i receives from TP_k after sending most recently a message to TP_j, and that M_{kj} are also destined for TP_j. TP_i sends m by the following procedure.

(1) $I_{kj} = M_{kj}$ if $\Delta_{ki} + \Delta_{ij} < \Delta_{kj}$ and $(1-\varepsilon_{ki}) \cdot (1-\varepsilon_{ij}) > (1-\varepsilon_{kj})$, otherwise ϕ.

Table 1 Delay [msec].

	Protocols	B	M	N		D
(1)	receipt(R)	376	376	377		376
	delivery(DL)	383	383	384		383
	rel. rec. (RR)	724	724	726		1128
(2)	detect (DT)	386	386	387	726*	762
	receipt (R)	1140	393	394	1103*	1135
	delivery (DL)	1141	394	395	1105*	1139
	rel. rec. (RR)	1527	735	736	1482*	1891

(2) TP_i sends m with $I_{1j} \cup \cdots \cup I_{nj}$ to TP_j. This is an example of a indirectly replicated protocol.

5. Evaluation of Protocols

5.1 Reliable Receipt

We evaluate the basic (B), modified (M), nested group (N), and decentralized (D) protocols in terms of the delay time for delivering and reliably receiving messages. Prototypes of the protocols were implemented as a group G of seven UNIX processes on SPARC workstations: three (*ktsun0*, *kelvin*, *ccsun*) in Hatoyama; one (*ipsj*) in Tokyo, Japan; two (*ucla*, *osu*) in the U.S.; and one (*des*) in Keele, UK. We consider two cases: (1) there is no message loss and (2) *kelvin* loses m. We measure the delay time when *des* in the UK sends a message m of 128 bytes to the three workstations in Hatoyama. In the B and D protocols, *des* retransmits m. In the M protocol, *ktsun0* the nearest to *kelvin*, forwards m to *kelvin*. In the N protocol, G is composed of the Keele and Hatoyama subgroups. The *Keele* subgroup consists of one workstation, *des*. In the *Hatoyama* subgroup of three workstations, *ktsun0* is the coordinator.

The following events occur in the process:

send: m is sent by the original sender process.

receive: m is received by the destination process.

deliver: m is delivered to an application process.

reliable receive: The sender process knows that m has been received by all the destinations.

detect: A destination process detects a loss of m by receiving another process's confirmation of m.

For each event e, let time(e) be the time at which e occurs. The following kinds of delays are obtained from the times measured:

receipt (R) delay: time(receive)−time(send).

delivery (DL) delay: time(deliver)−time(send).

reliable receipt (RR) delay: time(reliable receive)−time(send).

detect (DT) delay: time(detect)−time(send).

Part (1) of **Table 1** indicates the R, DL, and RR delays for four protocols in the first case. The difference between R and DL shows the time needed for the protocol processing. The difference between R and RR shows the time needed for exchanging the confirmation messages of m. Every protocol supports almost the same delay.

Part (2) of Table 1 shows the R, DT, DL, and RR delays in the presence of lost messages. The difference between DL and DT shows the time needed for recovering from the message loss by means of retransmission. For example, *ktsun0* forwards m to *kelvin* in the M protocol. The difference between DT and DL shows how long it takes to retransmit m. In the N protocol, we consider two cases: messages are lost between *des* to *ktsun0* and lost in Hatoyama. The delay times in the first case are marked * in Table 1.

As Table 1 shows, in the M protocol, the processes can recover from message loss with a shorter delay than in the other protocols. In addition, the delay time is almost the same as in the no-loss case. In the wide-area group, each channel has a different delay time and message loss ratio. Hence, the messages can be delivered with a shorter delay if they are sent through channels with a shorter delay and lower loss ratio.

5.2 Δ^*-Causality

Next, we evaluate protocols that provide G with Δ^*-causality in terms of the number of messages to be rejected. **Figure 6** shows the receipt ratio $R(t)$ (≤ 1) of messages sent to *kelvin* from *ipsj*, *ucla*, and *des* for the delay t, where $\int R(t)dt = 1$. The ratio is measured by transmitting 10,000 messages. For example, 2.7% of messages take around 120 msec to get to *ucla* from *kelvin*. **Table 2** shows the minimum, average, and maximum delays. In Hatoyama, there is no message loss and almost all mes-

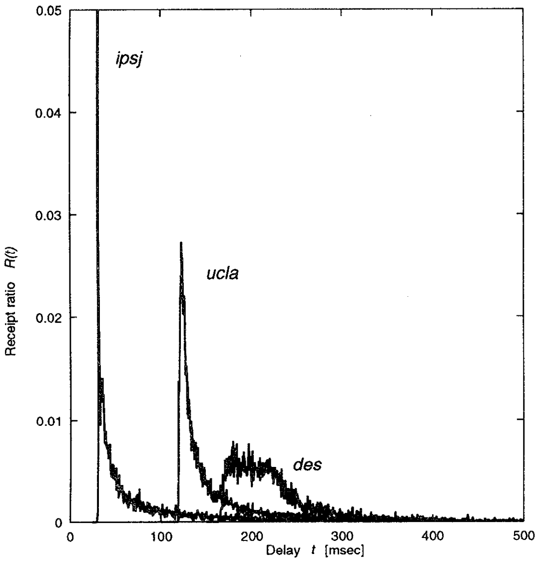


Fig. 6 Message receipt ratio vs. delay.

Table 2 Delay [msec] & lost [%].

Host	Min.	Avrg.	Max.	Lost
ipsj	30.437	60.427	756.263	0.9
ucla	119.506	157.171	532.433	8.3
des	164.497	241.370	2733.565	24.4

sages are received in 0.5 msec. Between Japan and the UK, on the other hand, one fourth of the messages are lost and transmission takes about 240 msec. The figure and table require that Δ_{ij} depend not only on δ_{ij} but also on ε_{ij}. In terms of the delay time and message loss, *osu* is almost the same as *ucla*.

Every TP_i obtains the statistics of the delay time δ_{ij} and the loss ratio ε_{ij} for each TP_j by using the GCM protocol. AP_i decides Δ_{ij} by using the statistics of δ_{ij} and ε_{ij}. One way to obtain Δ_{ij} is by adding the average δ_{ij} to some constant α_i. Another way is for Δ_{ij} to be given a time t within which a message can be received with a probability of β percent. For example, let β be 70%. From Fig. 6, 70% of messages sent by *ucla* can be received by *kelvin* in 168 msec. Hence, Δ_{ku} is given 168 msec. 70% of messages sent by *des* can be received in 320 msec. Hence, $\Delta_{kd} = 320$. On the other hand, the average delay time between *kelvin* and *ipsj* is about 60 msec, while only 0.1% of messages are lost. Here, let Δ_{ki} be 90, which is 50% larger than 60 msec; that is, $\alpha_i = 30\%$.

First, we consider how many messages each process can receive in a given Δ^*. As explained above, TP_i does not receive message m from

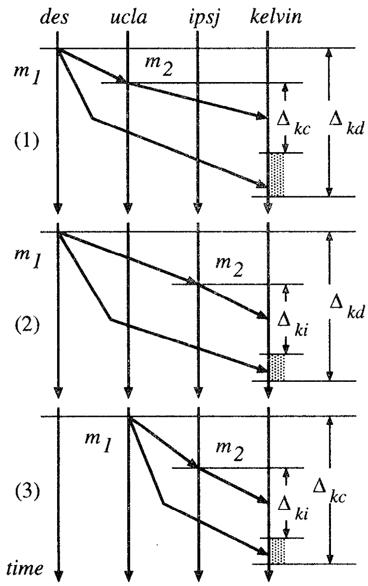


Fig. 7 Inconsistent Δ^*-causality.

Table 3 Message receipt ratio ε [%] in Δ^+.

	Δ [msec]	ipsj	ucla	des
Minimum	90	60.5	0.0	0.0
Average	168	94.5	70.0	7.6
Maximum	320	98.9	88.9	70.0

TP_j unless m arrives in Δ_{ij}. Given Δ^* presented here, 60.5% of messages sent by *ipsj* are received by *kelvin*. Hence, $\varepsilon_{ki} = 60.5\%$ for Δ_{ki}. Similarly, $\varepsilon_{ku} = 70.0$ and $\varepsilon_{kd} = 70.0$ for Δ_{ku} and Δ_{kd}, respectively.

Next, we consider the ratio of messages rejected due to the inconsistency of the Δ^*-causality. **Figure 7** shows how m_1 and m_2 such that $m_1 \overset{\Delta^*}{\rightarrow} m_2$ are received by *kelvin*. We assume $\delta_{de} = \delta_{kd} - \delta_{ku}$, $\delta_{di} = \delta_{kd} - \delta_{ki}$, and $\delta_{ci} = \delta_{ku} - \delta_{ki}$, since there is a routing path from Hatoyama via Tokyo and the USA to Keele. In Fig. 7, we also assume that *ucla* and *ipsj* send m_2 on receipt of m_1. In (1) and (2), m_1 is sent by *des*. In (3), *ucla* sends m_1. In (3), *ucla* sends m_2 on receiving m_1 while *ipsj* sends m_2 in (2) and (3). The reject ratios of messages received are 6.9% in (1), 16.6% in (2), and 16.7% in (3).

Next, suppose that Δ^+ is used, since Δ^* is inconsistent. Δ_i^+ can be the minimum, the average, or the maximum of $\Delta_{i1}, \cdots, \Delta_{in}$, which here are 90, 168, and 320 msec, respectively. **Table 3** shows the receipt ratios of messages sent to *kelvin* from *ipsj*, *ucla*, and *des*. For example, if $\Delta_k = 168$, *kelvin* receives 94.5% of messages from *ipsj*, while only 7.6% can be received from *des*.

If the Δ^*-causality is adopted, some messages are rejected to preserve the causality. On the other hand, in the Δ^+-causality, fewer messages are rejected; that is, the longer Δ is, the larger ε becomes but the longer it takes to deliver messages. More messages from the more distant processes are rejected, i.e. the shorter Δ is, the smaller ε is. Thus, there is a trade-off between Δ and ε. The application processes have to decide Δ so that the requirements regarding the delay time and causality are satisfied.

5.3 Message Buffering

In the B and D protocols, only the sender buffers m, since the sender retransmits m. Hence, the total number of buffers needed to store m in the group is 1. However, even if the B or D protocol is used, some messages are buffered in destinations, so that they can be ordered causally. In the M protocol, not only the sender but also the destination may retransmit m. Hence, the total number of buffers is $s+1$ for the number s of destinations. In the N protocol, the coordinators of the subgroups retransmit m. The total number of buffers is the number sg of subgroups if all subgroups include destinations. Hence, the protocol can be selected by considering the trade-off between the delay time and buffer space.

6. Concluding Remarks

We have discussed wide-area group communication, which includes multiple processes interconnected by the Internet. Here, each logical channel between the processes in a group has a different delay time and message loss ratio. We have presented ways of reducing the delay time of messages in a wide-area group, and discussed the Δ^*-causality in the wide-area group. We have also presented four kinds of protocols: basic, modified, nested group, and decentralized. Evaluation of these protocols in terms of the delay time shows that the modified protocol gives a shorter delay than the others.

Acknowledgments We would like to thank Prof. Lewis A. Davis, Tokyo Denki Univ. for his useful help and encouragement in writing this paper. We also would like to thank Prof. M. Liu, OSU, Prof. M. Gerla, UCLA, and Prof. S.M. Deen, Keele for their cooperation of the experiment.

References

1) Adelstein, F. and Singhal, M.: Real-Time Causal Message Ordering in Multimedia Systems, *Proc. IEEE ICDCS-15*, pp.36–43 (1995).
2) Aiello, R., Pagani, E. and Rossi, G.P.: Causal Ordering in Reliable Group Communications, *ACM. SIGCOMM '93*, pp.106–115 (1993).
3) Amir, Y., Dolev, D., Kramer, S. and Malki, D.: Transis: A Communication Sub-system for High Availability, *Proc.IEEE FTCS-22*, pp.76–84 (1993).
4) Amir, Y., Moser, L.E., Melliar-Smith, P.M. Agarwal, D.A. and Ciarfella, P.: The Totem Single-Ring Ordering and Membership Protocol, *Trans. ACM Computer Systems*, Vol.13, No.4, pp.311–342 (1995).
5) Baldoni, R., Mostefaoui, A. and Raynal, M.: Efficient Causally Ordered Communications for Multimedia Real-Time Applications, *Proc. IEEE HPDC-4*, pp.140–147 (1995).
6) Birman, K., Schiper, A. and Stephenson, P.: Lightweight Causal and Atomic Group Multicast, *ACM Trans. Computer Systems*, Vol.9, No.3, pp.272–314 (1991).
7) Chang, J.M. and Maxemchuk, N.F.: Reliable Broadcast Protocols, *ACM Trans. Computer Systems*, Vol.2, No.3, pp.251–273 (1984).
8) Cho, K. and Birman, K.: A Group Communication Approach for Mobile Computing, *Mobile Computing Workshop*, pp.95–102 (1994).
9) Ezhilchelvan, P.D., Macedo, R.A. and Shrivastava, S.K.: Newtop: A Fault-Tolerant Group Communication Protocol, *Proc. IEEE ICDCS-15*, pp.296–307 (1995).
10) Florin, G. and Toinard, C.: A New Way to Design Causally and Totally Ordered Multicast Protocols, *ACM Operating Systems Review*, Vol.26, No.4, pp.77–83 (1992).
11) Garcia-Molina, H. and Spauster, A.: Ordered and Reliable Multicast Communication, *ACM Trans. Computer Systems*, Vol.9, No.3, pp.242–271 (1991).
12) Hofmann, M., Braun, T. and Carle, G.: Multicast Communication in Large Scale Networks, *Third IEEE Workshop on High Performance Communication Subsystems (HPCS)*, (1995).
13) Holbrook, H.W., Singhal, S.K. and Cheriton, D.R.: Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation, *Proc. ACM SIGCOMM '95*, pp.328–341 (1995).
14) Jia, X.: A Total Ordering Multicast Protocol Using Propagation Trees, *IEEE Trans. Parallel and Distributed Systems*, Vol.6, No.6, pp.617–627 (1995).
15) Jones, M., Sorensen, S. and Wilbur, S.: Protocol Design for Large Group Multicast-

ing: The Message Distribution Protocol, *Computer Communications*, Vol.14, No.5, pp.287–297 (1991).

16) Kaashoek, M.F., Tanenbaum, A.S., Hummel, S.F. and Bal, H.E.: An Efficient Reliable Broadcast Protocol, *ACM Operating Systems Review*, Vol.23, No.4, pp.5–19 (1989).

17) Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System, *Comm. ACM*, Vol.21, No.7, pp.558–565 (1978).

18) Liang, L., Chan, S.T. and Neufeld, G.W.: Process Groups and Group Communications: Classifications and Requirements, *IEEE Computer*, Vol.23, No.2, pp.56–66 (1990).

19) Mattern, F.: Virtual Time and Global States of Distributed Systems, *Parallel and Distributed Algorithms*, Cosnard, M. and Quinton, P. (Eds.), pp.215–226, North-Holland (1989).

20) Melliar-Smith, P.M., Moser, L.E. and Agrawala, V.: Broadcast Protocols for Distributed Systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.1, No.1, pp.17–25 (1990).

21) Nakamura, A. and Takizawa, M.: Causally Ordering Broadcast Protocol, *Proc.IEEE ICDCS-14*, pp.48–55 (1994).

22) Peterson, L.L., Buchholz, N.C. and Ghemawat, S.: Preserving and Using Context Information in Interprocess Communication, *ACM Trans. Computer Systems*, Vol.7, No.3, pp.217–246 (1989).

23) Prakash, R., Raynal, M. and Singhal, M.: An Efficient Causal Ordering Algorithm for Mobile Computing Environments, *Proc.IEEE ICDCS-16*, pp.744–751 (1996).

24) Postel, J.: Internet Protocol, *RFC-791* (1981).

25) Postel, J.: Internet Control Message Protocol, *RFC-792* (1981).

26) Reiter, M.K.: The Rampart Toolkit for Building High-Integrity Services, *Theory and Practice in Distributed Systems*, Lecture Notes in Computer Science, Vol.938, pp.99–110, Springer-Verlag (1995).

27) Tachikawa, T. and Takizawa, M.: Selective Total Ordering Broadcast Protocol, *Proc. IEEE ICNP-94*, pp.212–219 (1994).

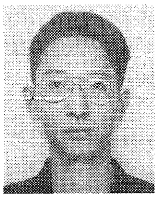
28) Tachikawa, T. and Takizawa, M.: Multimedia Intra-Group Communication Protocol, *Proc. IEEE HPDC-4*, pp.180–187 (1995).

29) Tachikawa, T. and Takizawa, M.: Distributed Protocol for Selective Intra-group Communication, *Proc. IEEE ICNP-95*, pp.234–241 (1995).

30) Takamura, A., Takizawa, M. and Nakamura, A.: Group Communication Protocol for Large Groups, *Proc. IEEE Conf. on Local Computer Networks (LCN-18)*, pp.310–319 (1993).

31) Yavatkar, R.: MCP: A Protocol for Coordination and Temporal Synchronization in Multimedia Collaborative Applications, *Proc. IEEE ICDCS-12*, pp.606–613 (1992).

(Received January 20, 1997)
(Accepted March 7, 1997)



Takayuki Tachikawa was born in 1971. He received his B.E. and M.E. degrees in computers and systems engineering from Tokyo Denki University, Japan in 1994 and 1996, respectively. Currently, he is studying towards the Ph.D. degree at computers and systems engineering, Tokyo Denki University. His research interests include distributed systems, computer networks, and communication protocols. He is a student member of IPSJ.

Makoto Takizawa was born in 1950. He received his B.E. and M.E. degrees in Applied Physics from Tohoku University, Japan, in 1973 and 1975, respectively. He received his D.E. in Computer Science from Tohoku University in 1983. From 1975 to 1986, he worked for Japan Information Processing Developing Center (JIPDEC) supported by the MITI. He is currently a Professor of the Department of Computers and Systems Engineering, Tokyo Denki University since 1986. From 1989 to 1990, he was a visiting professor of the GMD-IPSI, Germany. He is also a regular visiting professor of Keele University, England since 1990. He was a vice-chair of IEEE ICDCS, 1994, and is serving as a program co-chair of IEEE ICDCS, 1998 and serves on the program committees of many international conferences. His research interest includes communication protocols, group communication, distributed database systems, transaction management, and groupware. He is a member of IEEE, ACM, IPSJ, and IEICE.