Proceedings of International Conference
on Networking and Multimedia

Communication Protocol for Wide-area Group

Takayuki Tachikawa and Makoto Takizawa

Dept. of Computers and Systems Engineering
Tokyo Denki University
Ishizaka, Hatoyama, Hiki-gun, Saitama 350-03, JAPAN
e-mail {tachi, taki}@takilab.k.dendai.ac.jp

Abstract

Distributed systems are composed of multiple computers interconnected by communication networks. Group communication protocol supports the ordered and reliable delivery of messages to multiple destinations in a group of processes. The group communication protocols discussed so far 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 communication where the delay times among the processes are significantly different. We define newly a Δ^*-causality to be applied in the wide-area group. We present a protocol which supports a group of processes with the Δ^*-causality.

1 Introduction

Distributed systems are composed of multiple computers interconnected by communication networks. In distributed applications like teleconferences, a group of multiple processes, i.e. process group [16] have to be cooperated. Group communication protocols support a group of processes with the reliable and ordered delivery of messages to multiple destinations. Transis [2], ISIS(CBCAST) [5], Psync [20], and others [19,26] support the causally ordered delivery. Totem [3], ISIS(ABCAST) [5], Ameoba [14], Trans/Total [18], Rampart [23] and others [6, 24] support the totally ordered delivery. Systems [7,8] support both.

Group communication protocols discussed so far assume that every two processes have almost the same communication delay time. Here, let us consider a world-wide teleconference among processes K, C, T, and H in Keele of UK, Colorado in the USA, Tokyo, and Hatoyama of Japan, respectively. By using the Internet, it takes about 60 msec to propagate a message in Japan while taking about 240 msec between Tokyo and Europe. In addition, the longer the distance is, the more messages are lost. For example, about 20% of the messages are lost between Japan and Europe while less than 1% is lost in Japan. Thus, it is essential to consider a group communication where the delay times between the processes in the group are significantly different [10, 11, 13], i.e. not neglectable compared with the processing speed. Such a group of processes is named a *wide-area* group. If the traditional group communication protocols are adopted to the wide-area group, the time for delivering messages to the destinations is dominated by the largest delay. For example, if T sends a message m to H and K, T has to wait for the response from K while having received earlier the 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 m. The delay time between T and K is about four times longer than T and H. If H resends m, the time for retransmitting m can be reduced. Thus, we try to reduce the delivery time by the *destination* retransmission.

Suppose that T sends m to H, C, and K. On receipt of m, the destination processes send the receipt confirmation messages to T. Here, let us consider a way that K sends the confirmation to C instead of directly sending to T and then C sends the confirmation back to T. Even if C loses m, the delay time can be reduced if K retransmits m to C as presented before. A wide-area group G can be decomposed into disjoint subgroups G_1, ..., G_{sg} ($sg \geq 2$) [10, 27] where each G_i includes processes nearer to each other and has one coordinator process. Messages sent by a process are exchanged by the coordinators of the subgroups. In [11], each subgroup has a log to retransmit message.

In multimedia and realtime applications, messages have to be delivered in some predetermined time units. The Δ-causality [1, 4, 28] is discussed where Δ denotes the maximum delay time between the processes required by the application. That is, it is meaningless to receive a message m unless m is delivered in Δ after m is transmitted. The Δ-causality assumes that every 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 neglectable. In this paper, we newly define a Δ^*-*causality* where some pair of Δ_{ij} and Δ_{kl} are significantly different. Then, we present a protocol which supports the Δ^*-causality and can reduce the delay time and the number of messages with retransmissions by the destination and the group decomposition.

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.

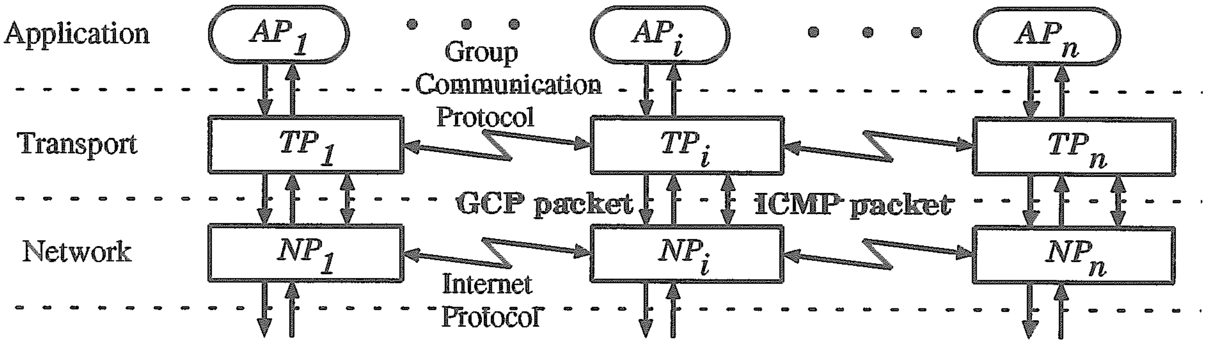

Figure 1: Distributed system

2 System Model

A distributed system is composed of three hierarchical layers, i.e. *application*, *transport*, and *network* layers as shown in Figure 1. A group of n (≥ 2) application processes AP_1, ..., AP_n are cooperated. Each AP_i communicates with other processes in the group by using the underlying group communication service provided by transport processes TP_1, ..., TP_n. Here, let G denote a group of the transport processes ($G = \{ TP_1, ..., TP_n \}$) supporting AP_1, ..., AP_n. G is considered to support each pair of processes TP_i and TP_j with a logical channel by using the underlying network layer. Data units transmitted at the transport layer are referred to as *packets*. TP_i sends a packet to TP_j by the channel. The network layer provides the IP service [21] for the transport layer.

The cooperation of the processes at the transport layer is coordinated by *group communication* (GC) and a *group communication management* (GCM) protocols. The GC protocol establishes a group G and reliably and causally [5] 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. In this paper, packets decomposed from the application message are referred to as *messages*.

A transport process TP_i has to know the delay time δ_{ij} with each TP_j in the group G. In the GCM protocol, TP_i requests the network layer to transmit two kinds of ICMP [22] packets: "Timestamp" and "Timestamp Reply". TP_i can know when "Timestamp" sent by TP_i is received by TP_j, and "Timestamp Reply" received by TP_i is sent by TP_j by using the time information. TP_i calculates δ_{ij} between TP_i and TP_j, i.e. round trip time. TP_i sends periodically 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 processes 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 which are not included in V.
- Packets sent by the same process may be received by the destination processes not in the sending order (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 $G = \{ TP_1, ..., TP_n \}$. m has to be *reliable* delivered to all the destinations in G. Here, let s be the number of the destinations of m. There are two points to be discussed to realize the reliable receipt of m:

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

In (1), there are two ways: *direct* and *hierarchical*. In the direct multicast, TP_i sends m directly to all the destinations. In the hierarchical multicast, TP_i sends m to a subset of the destinations. On receipt of m from TP_i, TP_j forwards m to other destinations. The propagation-tree based routing algorithms [9, 12] are discussed so far. Another example is to decompose G into disjoint subgroups G_1, ..., G_{sg} ($sg \geq 2$) [27]. 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.

In (2), there are two schemes: *decentralized* and *distributed*. In the decentralized scheme [5], TP_i sends m to the destinations and the destinations send back the receipt confirmation of m to TP_i. If TP_i receives all the confirmations, TP_i informs all the destinations of the reliable receipt of m. Totally $3s$ messages are transmitted and it takes three rounds where s is the number of destinations in G.

In the distributed scheme [24, 26], 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 the confirmations from all the destinations, TP_j reliably receives m. Here, $O(s^2)$ messages are transmitted and it takes two rounds. In [26], the number of messages transmitted in the group can be reduced to $O(s)$ by adopting the *piggy back* and the *deferred confirmation*.

There are the following protocols 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 [19]. The second is adopted by ISIS [5] and others [2,14,18].

Next, we consider when each destination process can deliver messages received. Here, let m_1 be a message received by TP_i. TP_i can deliver m_1 if (1) TP_i had 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 group G. Hence, the delay in delivering messages is increased if G includes more distant processes. Since the processes are assumed to be not 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, m can be retransmitted by TP_j.
(2) m can be removed from the buffer if m is reliably received, i.e. no need to retransmit m.

Hence, only the sender or process to retransmit m needs to know whether or not m is reliably received.

3.2 Recovery 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 the message loss. Let us consider a group H = { TP_1, TP_2, TP_3, TP_4 }. Figure 2 shows a *process graph* of H where each node denotes a process and each edge shows a channel between nodes. The weight of the channel ⟨ TP_i, TP_j ⟩ indicates the average delay time δ_{ij}. In Figure 2(2), a direct edge $TP_i \rightarrow TP_j$ means that TP_j is the nearest to TP_i. Suppose that TP_1 sends a message m to TP_2, TP_3, and TP_4, but TP_4 fails to receive m. In the traditional protocols, the sender TP_1 retransmits m to TP_4 and it takes $2\delta_{14}$. On the other hand, if TP_3 forwards m to TP_4, it takes $2\delta_{34}$. Since $\delta_{14} > \delta_{34}$, we can reduce time for retransmission of m if TP_3 forwards m to TP_4. Thus, if TP_j loses m, a process TP_k whose δ_{kj} is the minimum in G can send m to TP_j. As presented here, there are two ways to retransmit m if TP_j loses 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.

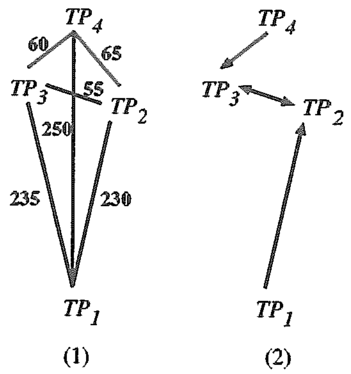

Figure 2: Process graph of the group H

3.3 Protocols

Suppose that a process TP_i sends m to a subset B of the destination processes in the group G. There are the following protocols.

(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.

[Basic(B) protocol]

(T1) TP_i sends m to all the processes in V ($\subseteq G$).
(T2) On receipt of m, each process TP_j in V sends the receipt confirmation to TP_i.
(T3) On receipt of the confirmation messages from all the processes in V, 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 the B one 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, the first step (T1) is executed again. □

In the third protocol, G is decomposed into disjoint subgroups G_1, ..., G_{sg} ($sg \geq 2$). Each G_i is composed of the processes TP_{i1}, ..., 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 a set of the 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 decentralized (D) protocol, only sender TP_i can know whether each destination receives m or not. Hence, the sender retransmission is adopted. In the D protocol, T1 and R are the same as the B protocol.

[Decentralized(D) protocol]

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

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

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

Figure 3(1), (2), and (4) show the B, M, and D protocols where TP_1 sends a message m to TP_2, TP_3, and TP_4 but TP_4 loses m. In the M protocol, TP_3 forwards m to TP_4 since TP_3 is the nearest to TP_4. Figure 3(3) shows the N protocol with two subgroups $\langle TP_{11}, TP_{12}, TP_{13} \rangle$ and $\langle TP_{21}, TP_{22}, TP_{23} \rangle$ where TP_{11} and TP_{21} are the coordinators. TP_{12} sends m but TP_{23} loses m. Here, TP_{21} resends m to TP_{23}.

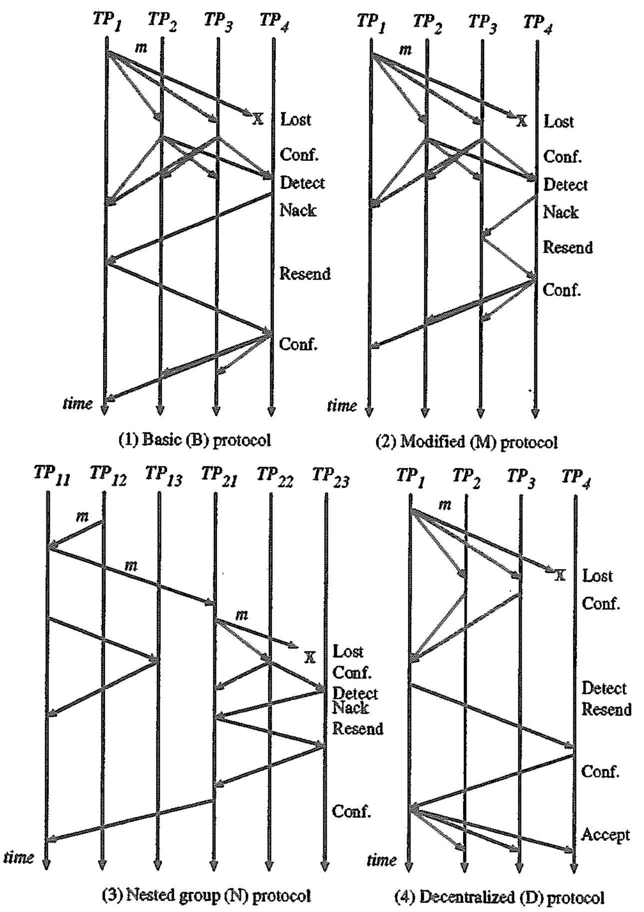

Figure 3: Protocols

3.4 Example

Let us consider the process graph shown in Figure 2. If TP_2 loses m and δ_{23} is the minimum, TP_3 forwards m to TP_2 in the modified (M) protocol. In the basic (B) one, TP_1 resends m to TP_2. In the nested group (N) one, H is composed of two subgroups $H_1 = \{ TP_1 \}$ and $H_2 = \{ TP_2, TP_3\ TP_4 \}$. TP_2 is the coordinator in H_2.

We now consider how long it takes to deliver m from TP_1 to TP_2, TP_3, and TP_4. Suppose that $\delta_{23} = \delta_{34} = \delta_{42}\ (= \delta_2) < \delta_{12} = \delta_{13} = \delta_{14}\ (= \delta_1)$. Here, let π_{ij} be a probability that a message sent by TP_i is lost by TP_j, i.e. $1 - \varepsilon_{ij}$. As shown before, the longer the distance, the more packets are lost. Hence, $\pi_{ij} \leq \pi_{kh}$ if $\delta_{ij} \leq \delta_{kh}$. We assume that $\pi_{12} = \pi_{13} = \pi_{14} = \pi_1$ and $\pi_{23} = \pi_{34} = \pi_{24} = \pi_2$. We also assume that no confirmation message is lost though messages may be lost. As presented before, the message loss ratios in Japan and between Japan and Europe are about 1% and 20%, respectively. Hence, we assume $\pi_1 > \pi_2$, $\pi_1^2 \simeq \pi_2$, and π_1^3 and π_2^2 can be neglected. We would like to show the average delay times D_B, D_M, D_N and D_D for the B, M, N, and D protocols, respectively.

$$D_B = (1 + 2\pi_1 + 2\pi_1^2)\delta_1.$$
$$D_M = \delta_1 + 2\pi_1(1 + \pi_2)\delta_2.$$
$$D_N = (1 + 2\pi_1 - 3\pi_1^2)\delta_1 + (2 + 4\pi_2 - \pi_1^2)\delta_2.$$
$$D_D = (2 + 2\pi_1 + 2\pi_1^2)\delta_1.$$

Here, suppose that $\delta_{12} = \delta_{13} = \delta_{14} = 240$, $\delta_{23} = \delta_{24} = \delta_{34} = 60$ [msec], $\pi_{12} = \pi_{13} = \pi_{14} = 0.2$ and $\pi_{23} = \pi_{24} = \pi_{34} = 0.01$. $D_B = 355$, $D_M = 264$, $D_N = 427$, and $D_D = 595$ [msec]. The M protocol implies the shortest delay.

In the B and D protocols, only TP_1 is required to buffer m since TP_1 retransmits m. All the processes may retransmit m. Hence, every process has to buffer m. In the N protocol, TP_1 sends m only to TP_2, and then TP_2 forwards m to TP_3 and TP_4. If either TP_3 or TP_4 loses m, TP_2 retransmits m. Hence, the coordinators have to have buffers. The B and D protocols imply a smaller number of buffers than the others.

4 Δ*-Causality

4.1 Δ-causality

The messages sent in a group $G = \{ TP_1, ..., TP_n \}$ have to be delivered in the causal order [5].

[Causal precedence relation] For every pair of messages 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 delivering a 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 [17].

In real time applications like multimedia communications, messages have to be delivered to the destinations by a deadline. Thus, a process TP_j has to receive a message m in Δ time units after TP_i sends m [1,4,28]. Δ denotes the maximum delay time between the processes in G. Here, let $ts(m)$ be time when m is sent. Let $tr_i(m)$ be time when TP_i receives m. Suppose that TP_i sends a message m to TP_j. 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 ($m_1 \xrightarrow{\Delta} m_2$) iff
(1) $m_1 \rightarrow m_2$ and (2) $ts(m_1) + \Delta \geq ts(m_2)$. □

Let us consider a group K = { TP_1, TP_2, TP_3 } as shown in Figure 4. Here, TP_1 sends a message m_1 to TP_2 and TP_3. TP_2 sends m_2 after receiving m_1 in Δ ($m_1 \xrightarrow{\Delta} m_2$). 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.

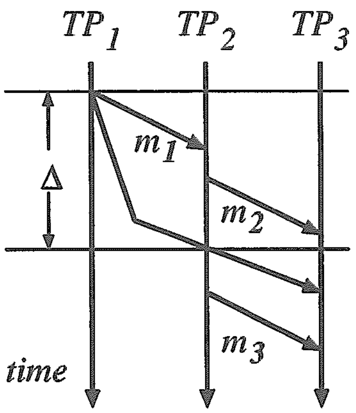

Figure 4: Δ-causality

4.2 Δ^*-causality

In a wide-area group G = { TP_1, ..., TP_n }, some pairs of delay times are significantly different. Here, let Δ_{ij} be obtained based on the statistics of δ_{ij} between TP_i and TP_j. For example, Δ_{ij} may be an average of δ_{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 { Δ_{ij} | $i, j = 1, ..., n$ }.

[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 ($m_1 \xrightarrow{\Delta^*} m_2$) iff
(1) $m_1 \rightarrow m_2$ and (2) $ts(m_1) + \Delta_{ij} \geq ts(m_2)$. □
That is, m_2 is sent in Δ_{ij} time units after m_1 is sent.

In Figure 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 is already delivered and $m_1 \xrightarrow{\Delta^*} m_2$, TP_3 cannot deliver m_1. If m_1 is delivered, m_2 cannot be delivered because m_2 is obligated to be delivered after $ts(m_2) + \Delta_{32}$. There is inconsistency among 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 $\xrightarrow{\Delta^*}$ is *consistent* iff for every pair of messages m_1 and m_2

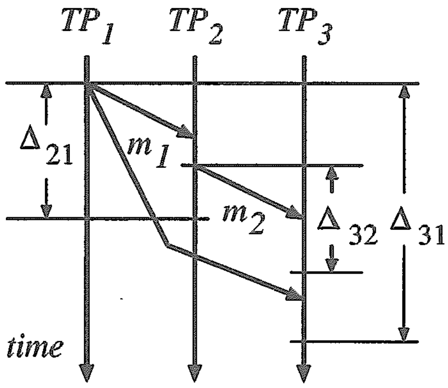

Figure 5: Δ^*-causality

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 is straightforward that the following theorem holds.

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

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

That is, the Δ-causality $\xrightarrow{\Delta}$ 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, i.e. Δ^* may be inconsistent. There are the following ways to resolve the inconsistency on the Δ^*-causality:

(1) to neglect messages which do not satisfy the Δ^*-causality, and
(2) to change some Δ_{ij} so that Δ^* is consistent.

First, let us consider the example of Figure 5. TP_3 receives m_2 in Δ_{23} and m_1 in Δ_{13}. One way is that TP_3 delivers m_2 just on Δ_{23} after m_2 is sent, i.e. 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, if m_1 is lost, neither m_1 nor m_2 is received although m_2 can be received.

For each TP_i, suppose that $min(\Delta_{1i}, ..., \Delta_{ni}) \leq \Delta_i \leq max(\Delta_{1i}, ..., \Delta_{ni})$. 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 gets, the more messages from the more distant processes are rejected.

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 a following set Δ^+ from Δ^*.

- Δ^+ = { Δ^+_{ji} | $\Delta^+_{ji} = \Delta_{ji}$ 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 $\stackrel{\Delta^+}{\rightarrow}$ 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. Therefore, we adopt the first way. In our implementation, we assume that realtime data is more often exchanged between processes which are nearer to each other. Hence, Δ_i is $min(\Delta_{1i}, ..., \Delta_{ni})$.

5 Evaluation of Protocols

5.1 Reliable receipt

Next 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. The prototypes of the protocols have been implemented to be a group G of six UNIX processes in SPARC workstations, i.e. three (*ktsun0*, *kelvin*, *ccsun*) in Hatoyama, one (*ipsj*) in Tokyo, Japan, one (*colorado*) 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 where *des* in UK sends a message m of 128 bytes to three workstations in Hatoyama. In the B and D protocols, *des* retransmits m. In the M protocol, *ktsun0* nearest to *kelvin* forwards m to *kelvin*. In the N protocol, G is composed of Keele and Hatoyama subgroups. The *Keele* subgroup includes one workstation *des*. In the *Hatoyama* subgroup including 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 is 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 time when 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).

(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 time for the protocol processing. The difference between R and RR shows time for exchanging the confirmation messages of m. Every protocol supports almost the same delay.

(2) of Table 1 shows the R, DT, DL, and RR delays in a case of lost messages. The difference between DL and DT shows time for recovering from the message loss by retransmission. For example, *ktsun0* forwards m to *kelvin* in the M protocol. The difference between DT and DL show how long it takes to retransmit m.

In the N protocol, we consider two cases: messages are lost from *des* to *ktsun0* and lost in Hatoyama. The delay times in the first case are marked * in Table 1.

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

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

5.2 Δ^*-causality

Next, we evaluate protocols which provide G with the 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*, *colorado*, and *des* for the delay t where $\int R(t)dt = 1$, which is measured by transmitting 10,000 messages. For example, 2.7% of messages take around 120 msec to get to *colorado* from *kelvin*. Table 2 shows the minimum, average, and maximum delays. In Hatoyama, there is no message loss and almost all messages are received in 0.5 msec. On the other hand, one fourth of the messages are lost and it takes about 240 msec between Japan and UK. The figure and table require that Δ_{ij} depend not only on δ_{ij} but also on ε_{ij}.

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

host	min	avrg	max	lost
ipsj	30.437	60.427	756.263	0.9
colorado	119.506	157.171	532.433	8.3
des	164.497	241.370	2733.565	24.4

Every TP_i gets the statistics of the delay time δ_{ij} and the loss ratio ε_{ij} for each TP_j by using the GCM protocol, and reports it to the application process AP_i. 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} with some constant α_i. Another way is for Δ_{ij} to be given time t within when a message can be received in a possibility of β percent. For example, let β be 70[%]. From Figure 6, 70% of messages sent by *colorado* can be received by *kelvin* in 168 msec. Hence, Δ_{kc} is given 168[msec]. 70% of messages sent by *des* can be received in 320 msec. Hence, Δ_{kd} =

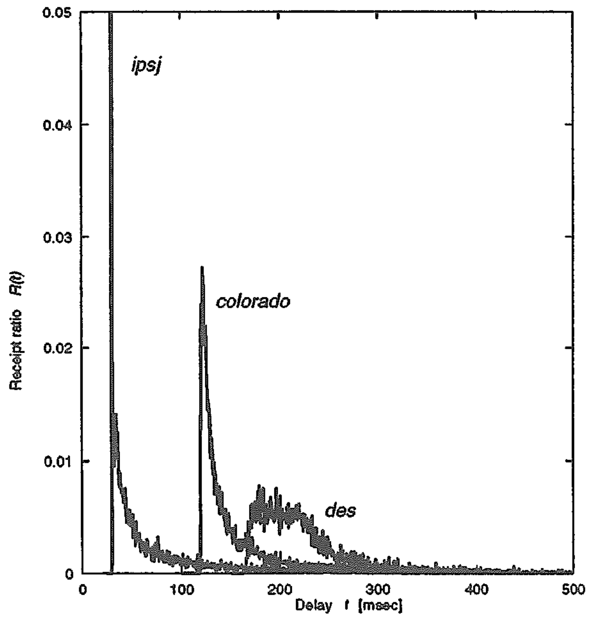

Figure 6: Message receipt ratio v.s. delay

320. On the other hand, the average delay time between *kelvin* and *ipsj* is about 60 msec where only 0.1% of messages are lost. Here, let Δ_{ki} be 90 which is 50% larger than 60 msec, i.e. $\alpha_i = 30\%$.

First, we consider how many messages each process can receive given Δ^*. As presented before, TP_i does not receive messages m from 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, $\varepsilon_{kc} = 70.0$ and $\varepsilon_{kd} = 70.0$ for Δ_{kc} 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 $\delta_{de} = \delta_{kd} - \delta_{kc}$, $\delta_{di} = \delta_{kd} - \delta_{ki}$, and $\delta_{ci} = \delta_{kc} - \delta_{ki}$ since there is a routing path from Hatoyama via Tokyo and Colorado to Keele. In Figure 7, we also assume that *colorado* and *ipsj* send m_2 on receipt of m_1. In (1) and (2), m_1 is sent by *des*. In (3), *colorado* sends m_1. In (3), *colorado* 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 the Δ^*-causality is inconsistent. Δ_i^+ can be either minimum, average, or maximum of Δ_{i1}, ..., Δ_{in}. Here, the minimum, average, and maximum are 90, 168, and 320 msec, respectively. Table 3 shows the receipt ratios of messages sent to *kelvin* from *ipsj*, *colorado*, 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, there exist some messages rejected to preserve the causality. On the other hand, in the Δ^+-causality, fewer messages are

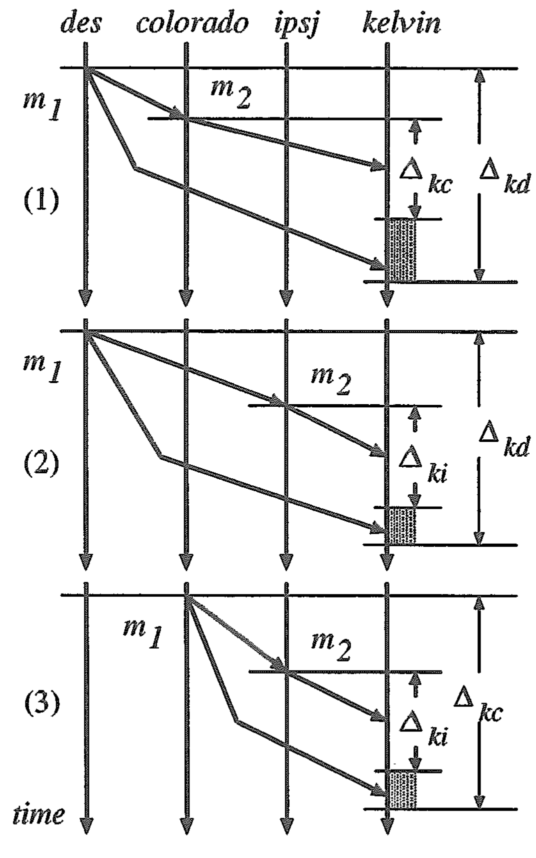

Figure 7: Inconsistent Δ^*-causality

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

	Δ [msec]	ipsj	colorado	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

rejected, i.e. the longer Δ is, the larger ε gets 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 on the delay time and the causality are satisfied.

6 Concluding Remarks

We have discussed the wide-area group communication which includes multiple processes interconnected by the Internet. Here, each logical channel between the processes in the group has a different delay time. In this paper, we have presented ways to reduce the delay time of messages in the wide-area group. In addition, we have discussed the Δ^*-causality in the wide-area group. We have presented four kinds of protocols, i.e. basic, modified, nested group, and decentralized protocols. These protocols have been evaluated in terms of the delay time. The evaluation shows that the modified protocol implies shorter delay than the others.

Acknowledgment

We would like to thank Prof. Lewis A. Davis, Tokyo Denki Univ. for his kind and useful help and encouragement in writing this paper.

References

[1] Adelstein, F. and Singhal, M., "Real-Time Causal Message Ordering in Multimedia Systems," *Proc. of IEEE ICDCS-15*, 1995, pp.36–43.

[2] Amir, Y., Dolev, D., Kramer, S., and Malki, D., "Transis: A Communication Sub-System for High Availability," *Proc. of IEEE FTCS-22*, 1993, pp.76–84.

[3] Amir, Y., Moser, L. E., Melliar-Smith, P. M. Agarwal, D. A., and Ciarfella, P., "The Totem Single-Ring Ordering and Membership Protocol," *Trans. on ACM Computer Systems*, Vol.13, No.4, 1995, pp.311–342.

[4] Baldoni, R., Mostefaoui, A., and Raynal, M., "Efficient Causally Ordered Communications for Multimedia Real-Time Applications," *Proc. of IEEE HPDC-4*, 1995, pp.140–147.

[5] Birman, K., Schiper, A., and Stephenson, P., "Lightweight Causal and Atomic Group Multicast," *ACM Trans. Computer Systems*, Vol.9, No.3, 1991, pp.272-314.

[6] Chang, J. M. and Maxemchuk, N. F., "Reliable Broadcast Protocols," *ACM Trans. Computer Systems*, Vol.2, No.3, 1984, pp.251–273.

[7] Ezhilchelvan, P. D., Macedo, R. A., and Shrivastava, S. K., "Newtop: A Fault-Tolerant Group Communication Protocol," *Proc. of IEEE ICDCS-15*, 1995, pp.296–307.

[8] Florin, G. and Toinard, C., "A New Way to Design Causally and Totally Ordered Multicast Protocols," *ACM Operating Systems Review*, Vol.26, No.4, 1992, pp.77–83.

[9] Garcia-Molina, H. and Spauster, A., "Ordered and Reliable Multicast Communication," *ACM Trans. Computer Systems*, Vol.9, No.3, 1991, pp.242–271.

[10] Hofmann, M., Braun, T., and Carle, G., "Multicast Communication in Large Scale Netwoks," *Third IEEE Workshop on High Performance Communication Subsystems(HPCS)*, 1995.

[11] Holbrook, H. W., Singhal, S. K., and Cheriton, D. R., "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," *Proc. of ACM SIGCOMM'95*, 1995, pp 328–341.

[12] Jia, X., "A Total Ordering Multicast Protocol Using Propagation Trees," *IEEE Trans. Parallel and Distributed Systems*, Vol.6, No.6, 1995, pp.617–627.

[13] Jones, M., Sorensen, S., and Wilbur, S., "Protocol Design for Large Group Multicasting: The Message Distribution Protocol," *Computer Communications*, Vol. 14 No. 5, 1991 pp.287–297.

[14] 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, 1989, pp.5–19.

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

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

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

[18] Melliar-Smith, P. M., Moser, L. E., and Agrawala, V., "Broadcast Protocols for Distributed Systems," *IEEE Trans. on Parallel and Distributed Systems*, Vol.1, No.1, 1990, pp.17–25.

[19] Nakamura, A. and Takizawa, M., "Causally Ordering Broadcast Protocol," *Proc. of IEEE ICDCS-14*, 1994, pp.48–55.

[20] Peterson, L. L., Buchholz, N. C., and Ghemawat, S., "Preserving and Using Context Information in Interprocess Communication," *ACM Trans. on Computer Systems*, Vol.7, No.3, 1989, pp.217–246.

[21] Postel, J., "Internet Protocol," *RFC-791*, 1981.

[22] Postel, J., "Internet Control Message Protocol," *RFC-792*, 1981.

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

[24] Tachikawa, T. and Takizawa, M., "Selective Total Ordering Broadcast Protocol," *Proc. of IEEE ICNP-94*, 1994, pp.212–219.

[25] Tachikawa, T. and Takizawa, M., "Multimedia Intra-Group Communication Protocol," *Proc. of IEEE HPDC-4*, 1995, pp.180–187.

[26] Tachikawa, T. and Takizawa, M., "Distributed Protocol for Selective Intra-group Communication," *Proc. of IEEE ICNP-95*, 1995, pp.234–241.

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

[28] Yavatkar, R., "MCP: A Protocol for Coordination and Temporal Synchronization in Multimedia Collaborative Applications," *Proc. of IEEE ICDCS-12*, 1992, pp.606–613.