Regular Paper

Group Protocol for Exchanging Multimedia Objects in a Group

Kenichi Shimamura,† Katsuya Tanaka† and Makoto Takizawa†

In distributed applications such as teleconferences and teleclassrooms, a group of multiple processes cooperate, and messages exchanged among the objects are required to be causally delivered. In addition, the processes exchange kinds of multimedia data. Multimedia messages are longer than traditional messages and are structured. In this paper, we discuss new types of causal precedence relations among multimedia messages. We also discuss how to exchange multimedia messages in a group of multiple processes, and evaluate the protocol.

1. Introduction

In distributed applications, a group of multiple processes cooperate. Various kinds of group protocols [3),15)] have been discussed in the literature. In group communication, a *group* is first established among multiple processes and then messages sent by the processes are *causally* or *totally* delivered to the destination processes in the group [3),8)]. A message m_1 *causally precedes* another message m_2 if the sending event of m_1 *happens before* ($\prec$) the sending event of m_2 [6)]. In totally ordered delivery, even messages that one not causally preceded are delivered to every common destination of the messages in the same order. In the protocols, messages transmitted at the network level are ordered independently of the information that applications include in the messages.

In distributed applications, not only traditional text data but also various kinds of multimedia objects such as images and video sequences are exchanged among the processes in the group. Multimedia objects are larger and more complex and structured than the traditional data messages exchanged among the processes. Several papers [1),2),18)] discuss Δ-causality, where Δ is the maximum delay time in the system. Tachikawa, et al. [16)] define the Δ-ϵ causality among messages, where Δ_{st} is the maximum delay time which the application can take and ϵ_{st} is the maximum ratio of messages to be lost between every pair of processes p_s and p_t. They discuss how to retransmit messages so as to satisfy constraints such as Δ and ϵ even if some destination process fails to receive the messages.

† Department of Computers and Systems Engineering, Tokyo Denki University

The object o is decomposed into a sequence of messages. A message is a unit of data transmitted in the network. If a pair of objects o_1 and o_2 are transmitted by processes p_1 and p_2, respectively, the messages of o_1 and o_2 are causally delivered in every common destination process p_3 of o_1 and o_2 according to traditional group protocols [3)]. In an application, the messages of o_1 can be delivered independently of o_2, and o_1 and o_2 are manipulated independently. In another application, the top message of o_1 is required to be delivered before the top message of o_2, while the other messages can be delivered in any order. Thus, we define new types of precedence relations of messages based on the object concept. According to the precedence relations, the destination process delivers messages of objects to the application. A pair of messages that are not ordered in the precedence relations can be delivered in any order. We discuss a protocol that supports the various types of causal precedence relations, named the *multimedia causally ordered* (*MCO*) protocol.

In Section 2, we present a system model. In Section 3, types of causal precedence relations among multimedia objects are discussed. In Section 4, we present the MCO protocol for exchanging multimedia objects in a group of processes. In Section 5, we evaluate of the MCO protocol.

2. System Model

Distributed applications are realized by the cooperation of a group of application processes $A_1, \ldots, A_n$ ($n \geq 1$), which are interconnected in a *reliable synchronous* network. Application processes exchange messages including multimedia data with the other processes in the group by using the network. A unit of data exchanged among the processes is referred to as

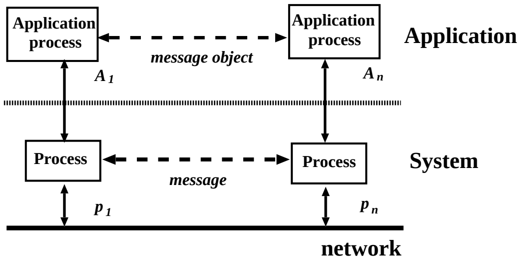


Fig. 1 Hierarchical structure of the system.

a *message object*, which use will fefer to simply as an *object*.

An application process A_t is supported by a system process p_t $(t = 1, \ldots, n)$ as shown in **Fig. 1**. A system process p_s takes an object from the application process A_s and then delivers the object to the system processes supporting the destination application processes by using the basic communication service supported by the network. In the remainder of the paper, we will use the term *process* to denote a system process. A data unit exchanged by the processes in the network is referred to as a *message*. In our system, we assume that the network supports processes with synchronous communication. That is, messages are not lost and the delay time between a pair of processes is bounded in the network. An object is decomposed into a sequence of messages and the messages are delivered to the destination processes. The destination process p_t assembles the messages received into an object and then delivers the object to the application process A_t. The cooperation of the processes supporting the group of the application processes is coordinated by a *group protocol* which supports the reliable, efficient communication service of multimedia objects by making use of the network service.

Multimedia objects are exchanged by the application processes. Suppose an object o is composed of three objects o_1, o_2, and o_3, which are referred to as *component* objects of o. The object o_i is also referred to as a *part* of o $(i = 1, 2, 3)$. The object o_2 is further composed of objects o_{21} and o_{22}. Messages carrying the object o finally include the lowest-level objects, i.e., leaf objects o_1, o_{21}, o_{22}, and o_3. The hierarchy of the objects in the *part-of* relation is referred to as an *object tree*. Parent, child, descendant, ancestor, root, and leaf in the object tree are defined according to the convention of the tree structure. Here, suppose the object

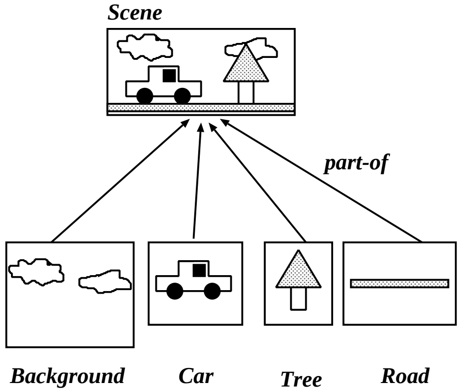


Fig. 2 Scene object.

o_1 is required to be displayed before o_2, and that o_2 is required to be displayed before o_3. Suppose o_{21} and o_{22} can be displayed in any order. The leaf objects o_1, o_{21}, o_{22}, and o_3 are ordered. The object o can be serialized into a sequence of leaf objects $\langle o_1, o_{21}, o_{22}, o_3\rangle$ or $\langle o_1, o_{22}, o_{21}, o_3\rangle$. The process takes the object o, say the sequence $\langle o_1, o_{21}, o_{22}, o_3\rangle$, and decomposes it into messages to be transmitted in the networks. We assume that each object is realized by a sequence of one or more messages and that each message includes data from at most one object.

A multimedia object is carried by messages, as explained in the preceding section. A multimedia object is furthermore composed of objects. Thus, the multimedia objects are hierarchically structured as shown in **Fig. 2**. For example, let us consider a *scene* of a video where a *car* is moving along a *road* and *trees* are seen from the *car*, as shown in Fig. 2. The *scene* object is composed of four component objects: *car*, *road*, *tree*, and *background*. In displaying the *scene* object in an application, the *road*, *tree*, and *background* objects are required to be displayed before the *car* object is displayed. Thus, the *car* object is preceded by the other objects.

3. Causality of Multimedia Objects

3.1 Traditional Messages

The *happen-before* relation ($\prec$) among events occurring in a distributed system is defined by Lamport [6]. The causal precedence relation among messages is defined as follows [6]:

- A message m_1 *causally precedes* another message m_2 iff a sending event of m_1 happens before ($\prec$) a sending event of m_2.

Figure 3 shows three processes p_s, p_t, and

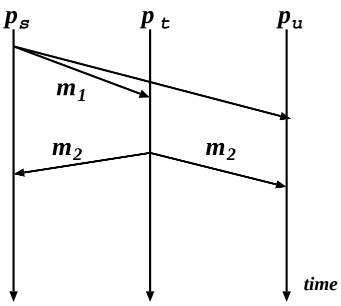


Fig. 3 Causal precedence of messages.

p_u exchanging messages. Process p_s sends a message m_1 to p_t and p_u. Process p_t sends a message m_2 to p_u after receiving m_1. Since the sending event of m_1 *happens before* the sending event of m_2, m_1 *causally precedes* m_2. Process p_u has to receive m_1 before m_2. In order to causally order the messages, the *vector clock* [8] is widely used in group protocols [3]. Suppose that there are n (>1) processes $p_1, \ldots, p_n$ in a group G. Each process p_t manipulates a vector clock $V = \langle V_1, \ldots, V_n \rangle$, where each element V_v is initially 0 for $v = 1, \ldots, n$. When p_t sends a message m, $V_t := V_t + 1$ and m carries the vector clock $m.V$ $(= V)$. On receipt of a message m, $V_u := \max(V_u, m.V_u)$ for $u = 1, \ldots, n$. For a pair of vectors $A = \langle A_1, \ldots, A_n \rangle$ and $B = \langle B_1, \ldots, B_n \rangle$, $A < B$ iff $A_i \leq B_i$ for every i and $A_j < B_j$ for every j. A message m_1 causally precedes another message m_2 iff $m_1.V < m_2.V$. The process p_u delivers m_1 before m_2 if $m_1.V < m_2.V$.

3.2 Multimedia Objects

Suppose a group G is composed of of processes $p_1, \ldots, p_n$ $(n > 1)$. Suppose that a process p_s sends an object o to another process p_t. Since a multimedia object is larger than a traditional message, it takes longer to send and receive the multimedia object. In order to increase the throughput and reduce the response time, the sending and receiving events of objects are interleaved if there is no precedent relation among the objects. That is, a process may send and receive messages of an object while the process is sending and receiving other objects.

Figure 4 shows three processes p_s, p_t, and p_u exchanging objects o_1 and o_2. In Fig. 4 (3), the process p_t starts to send messages of an object o_2 after receiving all the messages of another object o_1. According to the traditional causality theory [8], o_1 causally precedes o_2. In

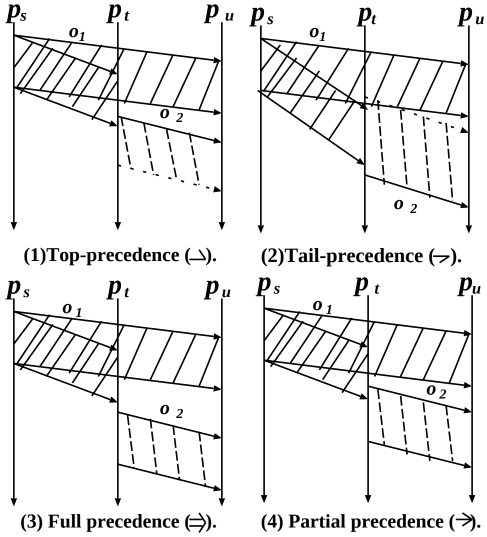


Fig. 4 Types of precedence of objects.

Fig. 4 (1), p_t starts to send a message of the object o_2 before receiving all the messages of o_1. Here, o_1 does not causally precede o_2. In Fig. 4 (2), p_t sends o_2 while receiving o_1. On the other hand, p_t sends o_2 after receiving all the messages of o_1. Here, o_1 does not causally precede o_2 either.

[Example 1] Let us consider an example of a teleconference where the participants are distributed among three remote sites S_s, S_t, and S_u. The teleconference is realized by a group of three processes p_s, p_t, and p_u, which support the sites S_s, S_t, and S_u, respectively, in Fig. 4. Each process performs the communication functions for each site. Participants in the conference share a virtual conference space C composed of three subspaces C_s, C_t, and C_u, each of which shows the participants attending at sites S_s, S_t, and S_u, respectively. The virtual space C is displayed at each site. Each site S_i distributes its subspace object C_i, which includes an image of the site, the voices of participants, and manuscripts to be handed out to all the processes in the group $(i = s, t, u)$. Suppose that some participant supported by the process p_s expresses some opinion which is represented by a voice and image object o_1. The process p_s distributes messages of the object o_1. After listening to the participant from p_s, a participant in p_t expresses a counter-opinion to o_1, which is carried by a multimedia object o_2. Here, the process p_u receives messages of the objects o_1 and o_2. The process p_t starts to send o_2 after receiving all the messages of o_1. Hence, p_u has to receive o_2 after o_1, as shown in Fig. 4 (3).

Next, suppose that some participant supported by the process p_s is expressing an opinion that is represented by an object o_1. While listening to the participant from p_s, a participant in p_t is leaving the conference. An image object o_2 showing his leaving the conference is distributed to the group. The process p_u has to start to deliver o_2 after starting to deliver o_1, as shown in Fig. 4 (1).

Suppose that the process p_s sends a music object o_1 which indicates that the conference place will be closed soon. The music stops only after every participant has left the conference. The process p_t sends an object showing the participants. Hence, p_u has to deliver o_2 before finishing delivering o_1, as shown in Fig. 4 (2). □

Following this example, the objects o_1 and o_2 are interrelated with respect to when the transmission of the messages is started in Fig. 4. We discuss how a pair of objects o_1 and o_2 can be causally preceded. Let $ss_t(o)$ and $es_t(o)$ denote events where in p_t starts to send an object o and finishes sending o, respectively. Let $sr_t(o)$ and $er_t(o)$ denote events where in p_t starts and finishes receiving the object o, respectively. A pair of starting event $ss_t(o)$ and ending event $es_t(o)$ for sending a traditional object o occur simultaneously, and a pair of receipt events $sr_t(o)$ and $er_t(o)$ also occur simultaneously in a process. However, these events cannot be assumed to occur simultaneously in the communication of the multimedia objects.

[Definition] The following types of precedent relations are defined for a pair of objects o_1 and o_2 sent by processes p_s and p_t, respectively:

- o_1 *top-precedes* o_2 ($o_1 \rightharpoonup o_2$) iff
  - ◇ $sr_t(o_1)$ happens before ($\prec$) $ss_t(o_2)$ if $p_s \neq p_t$.
  - ◇ $ss_s(o_1) \prec ss_t(o_2)$ if $p_s = p_t$.
- o_1 *tail-precedes* o_2 ($o_1 \rightharpoondown o_2$) iff
  - ◇ $er_t(o_1) \prec es_t(o_2)$ and $ss_s(o_1) \prec es_s(o_2)$ if $p_s \neq p_t$.
  - ◇ $es_s(o_1) \prec ss_t(o_2)$ if $p_s = p_t$.
- o_1 *fully precedes* o_2 ($o_1 \Rightarrow o_2$) iff
  - ◇ $er_s(o_1) \prec ss_t(o_2)$ if $p_s \neq p_t$.
  - ◇ $es_s(o_1) \prec ss_t(o_2)$ if $p_s = p_t$. □

In Fig. 4, $o_1 \rightharpoonup o_2$ in (1), $o_1 \rightharpoondown o_2$ in (2), and $o_1 \Rightarrow o_2$ in (3). The process p_u is required to deliver the messages of objects o_1 and o_2 so that the causalities defined here are preserved. An object o_1 is *interleaved* with another object o_2 iff $ss_t(o_2)$ happens before $er_t(o_1)$ and $sr_t(o_1)$ happens before $ss_t(o_2)$ in a source process p_t of o_2. Here, the process p_t is receiving messages of the object o_1 and sending messages of o_2 in an interleaved manner.

- o_1 *partially precedes* o_2 ($o_1 \rightarrow o_2$) iff $o_1 \rightharpoonup o_2$, $o_1 \rightharpoondown o_2$, and o_1 is interleaved with o_2 (Fig. 4 (4)).

The top, tail, fully, and partially precedent relations are referred to as *object-causally precedent* relations.

The following properties hold for the types of the object causally precedent relations:

[Properties] The following relations on the objects o_1, o_2, and o_3 hold:

- $o_1 \Rightarrow o_3$ if $o_1 \Rightarrow o_2$ and $o_2 \Rightarrow o_3$.
- $o_1 \rightharpoonup o_3$ if $o_1 \rightharpoonup o_2$ and $o_2 \rightharpoonup o_3$.
- $o_1 \rightharpoondown o_3$ if $o_1 \rightharpoondown o_2$ and $o_2 \rightharpoondown o_3$.
- $o_1 \Rightarrow o_3$ if $o_1 \Rightarrow o_2$ and $o_2 \rightharpoonup o_3$.
- $o_1 \Rightarrow o_3$ if $o_1 \rightharpoondown o_2$ and $o_2 \Rightarrow o_3$.
- $o_1 \rightharpoonup o_2$ and $o_1 \rightharpoondown o_2$ if $o_1 \Rightarrow o_2$.
- $o_1 \Rightarrow o_2$ if $o_1 \rightarrow o_2$.
- $o_1 \rightharpoonup o_2$ and $o_1 \rightharpoondown o_2$ if $o_1 \rightarrow o_2$. □

The precedent relations $\Rightarrow$, $\rightharpoonup$, and $\rightharpoondown$ are transitive according to the definitions. Discussion is still continuing on whether or not the partial precedent relation $\rightarrow$ is transitive. Suppose that there are four processes p_s, p_t, p_u, and p_v (**Fig. 5**). Suppose that the process p_s sends an object o_1 to p_t, p_u, and p_v, the process p_t sends o_2 to p_v while receiving o_1, and the process p_u sends o_3 to p_v while receiving o_2. Here, suppose that o_1 partially precedes o_2 ($o_1 \rightarrow o_2$) and $o_2 \rightarrow o_3$. The process p_v receives o_1 and o_2 in an interleaved manner and also receives o_2 and o_3 in an interleaved manner. The problem is how the process p_v receives the objects o_1 and o_3. If $o_1 \rightarrow o_3$, p_v is required to receive o_1 and o_3 in an interleaved manner. Otherwise, p_v can receive o_3 after o_1. Let us consider a virtual conference including four remote sites supported by four processes p_s, p_t, p_u, and p_v, as shown in Fig. 5. Suppose a participant of p_s is giving a presentation to all the participants in

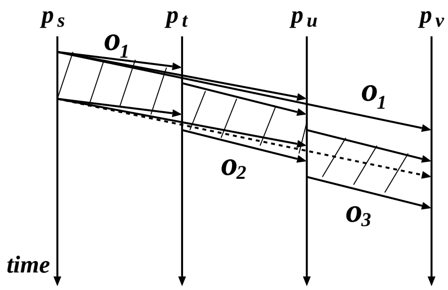


Fig. 5 Partially precedent relation of objects.

the conference. The presentation is realized by a multimedia object o_1 including voice, image, and pictures. Here, suppose some participant of p_t leaves the conference and information on his leaving is carried by an object o_2. A participant of p_u, who finds out that the participant of p_t has left the conference, expresses some opinion about his leaving to the chair of the conference of p_v. This proposal is carried by an object o_3. The chair of p_v is required to start to receive the object o_3 while receiving the object o_1. In this example, $o_1 \rightarrow o_3$, since $o_1 \rightarrow o_2$ and $o_2 \rightarrow o_3$; i.e., $\rightarrow$ is transitive. Thus, it depends on the applications whether or not the partially precedent relation $\rightarrow$ is transitive.

4. MCO Protocol

We present a multimedia causally ordered (*MCO*) protocol for supporting the object causally ordered delivery of multimedia objects for a group G which composed of multiple processes $p_1, \ldots, p_n$ $(n > 1)$.

4.1 Basic protocol

First, we discuss a basic protocol whereby each process sends an object at a time. An object o is decomposed into a sequence of messages. The first message in the sequence is referred to as the *top* message of the object o. Messages are preceded in the object causally precedent relations $\rightharpoonup$, $\rightharpoondown$, $\rightarrow$, and $\Rightarrow$ by using the vector clock $V = \langle V_1, \ldots, V_n \rangle$ with the bitmap $f = [f_1, \ldots, f_n]$. Each bit f_t shows whether or not a process p_t is sending an object. The vector clock V is manipulated in the same way in Mattern [8]. Each two elements V_t and f_t are defined for a process p_t; initially, $V_t = 0$ and $f_t = 1$ for $t = 1, \ldots, n$.

Suppose that a process p_t sends an object o. The variables V and f are manipulated in p_t as follows:

- $V_t := V_t + 1$;
- $f_t := 0$.

Only the top message m of the object o carries the vector V and f as $m.V$ and the bitmap $m.f$, respectively, to the destination processes in the group G. The messages of the object o do not carry V and f in order to reduce the communication overhead. V_t is incremented by 1 each time p_t starts to send an object. "$f_t = 0$" means that the process p_t is now sending an object. "$f_t = 1$" shows that p_t is not sending any object.

When it has finished sending the object o, the process p_t manipulates the variables as follows:

- $f_t := 1$:

The last message m of the object o carries $m.f_t$ (=1) to the destinations.

Next, suppose that a process p_t receives a message m of an object o from another process p_s. p_t manipulates the vector clock V and the bitmap f as follows:

- $V_s := \max\ (V_s,\ m.V_s)$ (for $s = 1, \ldots, n,\ s \neq t$);
- $f_s := 0$.

If the process p_t receives the whole object o from p_s, i.e. the last message m of the object o, p_t changes the bitmap f as follows:

- $f_t := 1$;

Let $o.sf$ and $o.ef$ show the bitmaps f which are carried by the top message and the last message of the object o, respectively. $o.f$ is used to show the current value of the bitmap f of the object o. $o.sf_s = 1$ and $o.ef_s = 0$ before the object o is transmitted by a process p_s. Let $o.SV$ and $o.EV$ show the vectors carried by the top and last messages of the object o, respectively. By using the vector clock V and the bitmap f, a process p_u orders the messages according to the following theorem.

[Theorem] Suppose that a process p_s sends an object o_1 and another process p_t sends an object o_2 to the other processes.

- $o_1 \Rightarrow o_2$ if $o_1.SV_v \leq o_2.SV_v$ $(v = 1, \ldots, n,\ v \neq s)$, $o_1.SV_s = o_2.SV_s$, and $o_2.sf_s = 1$
- $o_1 \rightharpoonup o_2$ if $o_1.SV_v \leq o_2.SV_v$ $(v = 1, \ldots, n,\ v \neq s)$ and $o_1.SV_s = o_2.SV_s$.
- $o_1 \rightharpoondown o_2$ if $o_1.EV_v \leq o_2.EV_v$ $(v = 1, \ldots, n,\ v \neq s)$ and $o_2.ef_s = 1$.
- $o_1 \rightarrow o_2$ if $o_1.SV_v \leq o_2.SV_v$ $(v = 1, \ldots, n,\ v \neq s)$, $o_1.SV_s = o_2.SV_s$, and $o_2.sf_s = 0$. □

[Example 2] Figure 6 shows an example in which three processes p_s, p_t, and p_u send and receive objects. Here, $\langle \ldots \rangle$ and $[\ldots]$ show the vector clock and the bitmap, respectively.

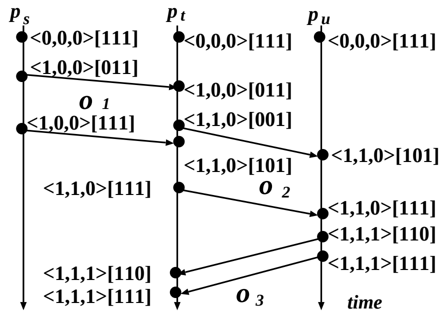


Fig. 6 Transmission of multimedia objects.

First, p_s starts sending an object o_1 to p_t and p_u, where $o_1.SV = \langle 1,\ 0,\ 0\rangle$ and $o_1.sf = [0\ 1\ 1]$. Since p_s is sending messages of the object o_1 to p_t and p_u, $o_1.f_t = o_1.f_u = 1$. Process p_s sends $o_2.ef = [1\ 1\ 1]$ if p_s finishes transmitting the object o_1. Process p_t starts sending an object o_2 to p_u before receiving all messages of o_1; i.e., $o_1 \rightharpoonup o_2$. Here, $o_2.SV = \langle 1,\ 1,\ 0\rangle$ and $o_2.sf = [0\ 0\ 1]$. Process p_u sends an object o_3 after receiving o_2; i.e., $o_2 \Rightarrow o_3$. Here, $o_2.SV = \langle 1,\ 1,\ 1\rangle$ and $o_2.sf = [1\ 1\ 0]$. □

In Example 2, $o_1 \Rightarrow o_3$, since $o_1 \rightharpoonup o_2$ and $o_2 \Rightarrow o_3$. Thus, this protocol can causally order o_1 and o_2 if o_1 is *directly* causally preceded by o_2. Even if o_2 transitively precedes o_1, the protocol cannot order o_1 and o_2.

4.2 Modified protocol

Next, we discuss a modified protocol whereby each process can send multiple objects at a time and objects can be transitively preceded. Two vectors of variables $V = \langle V_1, \ldots, V_n\rangle$ and $A = \langle A_1, \ldots, A_n\rangle$ instead of bitmaps are manipulated in a process. V is the vector clock. A is used to precede objects. Each pair of elements V_t and A_t are used for a process p_t. Each element A_t takes a integer value, not bit. Let $o.SA$ denote the value of A when the transmission of an object o is started, and let $o.EA$ show the value of A when the transmission of the object o.

Initially, $V = \langle 0, \ldots, 0\rangle$ and $A = \langle 0, \ldots, 0\rangle$. V and A are manipulated in a process p_t as follows each time p_t sends an object o:

- $V_t := V_t + 1$;
- $A_t := A_t + 1$.

The variable A is also incremented by 1 when p_t finishes sending the object o.

- $A_t := A_t + 1$;

However, V_t is not changed.

On receiving the top message of an object o from a process p_s, the process p_t manipulates the variables V and A as follows:

- $V_s := \max(V_s,\ o.SV_s)$ $(s = 1, \ldots, n, \neq t)$;
- $A_s := \max(A_s,\ o.SA_s)$ $(s = 1, \ldots, n, \neq t)$.

The following theorem holds from the definitions.

[Theorem] Suppose that a process p_s sends an object o_1 and another process p_t sends an object o_2 to the other processes.

- $o_1 \Rightarrow o_2$ iff $o_1.EA_v \leq o_2.SA_v$ $(v = 1, \ldots, n,\ v \neq s)$.
- $o_1 \rightharpoonup o_2$ iff $o_1.SV_v \leq o_2.SV_v$ $(v = 1, \ldots, n,\ v \neq s)$.
- $o_1 \rightharpoondown o_2$ iff $o_1.EA_v \leq o_2.EA_v$ $(v = 1, \ldots, n,\ v \neq s)$.
- $o_1 \rightarrow o_2$ iff $o_1.EA_v \geq o_2.SA_v$, $o_1.EA_v < o_2.EA_v$, and $o_1.SV_v \leq o_2.SV_v$ $(v = 1, \ldots, n,\ v \neq s)$. □

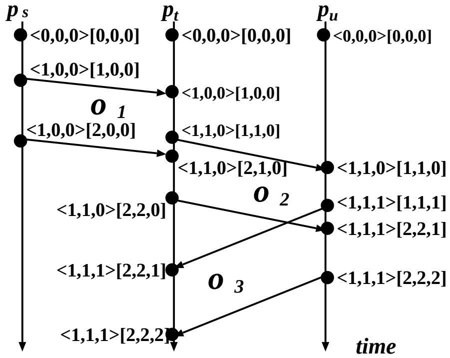


Fig. 7 Modified protocol.

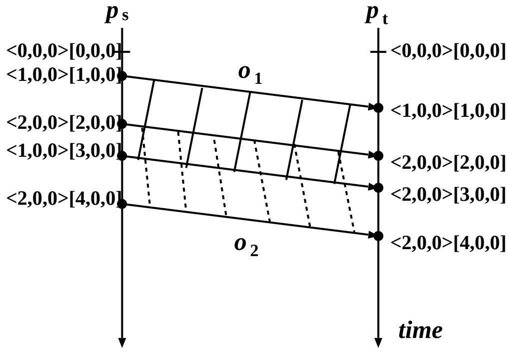


Fig. 8 Interleaving objects.

The objects received are ordered by using the vectors V and A according to the rules on the vectors presented in the theorem.

[Example 3] Figure 7 shows three processes p_s, p_t, and p_u which are exchanging objects o_1, o_2, and o_3. First, p_s starts to send o_1 to p_t and p_u. Here, $o_1.SV = \langle 1,\ 0,\ 0\rangle$ and $o_1.SA = [1,\ 0,\ 0]$. The process p_t starts sending o_2 while p_t is receiving the object o_1 from p_s, i.e. $o_1 \rightharpoonup o_2$. Here, $o_2.SV = \langle 1,\ 1,\ 0\rangle$ and $o_2.SA = [1,\ 1,\ 0]$. $o_1.SV < o_2.SV$. Then, the process p_u starts to send an object o_3 while receiving the object o_2. Here, $o_3.SV = \langle 1,\ 1,\ 1\rangle$ and $o_3.SA = [1,\ 1,\ 1]$. Since $o_1.SV < o_3.SV$, $o_1 \rightharpoonup o_3$. □

[Example 4] In **Fig. 8**, a process p_s sends objects o_1 and o_2 to p_t in an interleaved manner. When p_s starts sending o_1, $o_1.SV = \langle 1,\ 0,\ 0\rangle$ and $o_1.SA = [1,\ 0,\ 0]$. Then, p_s starts sending o_2, where $o_2.SV = \langle 2,\ 0,\ 0\rangle$ and $o_2.SA =$

[2, 0, 0]. When p_s finishes sending o_1, $o_1.EV = \langle 1, 0, 0 \rangle$ and $o_1.EA = [3, 0, 0]$. Process p_s finishes sending o_2, where $o_2.EV = \langle 2, 0, 0 \rangle$ and $o_2.EA = [4, 0, 0]$. Process p_t receives o_1 and o_2 from p_s. $o_1 \rightarrow o_2$, since $o_1.EA > o_2.SA$, $o_1.EA < o_2.EA$, and $o_1.SV < o_2.SV$. □

5. Evaluation

We evaluate the MCO group protocol discussed here in terms of the number of network-level messages to be causally ordered by comparing it with the traditional network-level causality. A process p_t sends messages to the processes and receives messages from the precesses in the group. Suppose that a process p_t receives messages $m_{21}, \ldots, m_{2l}$ after sending m_1 and before sending m_2 (**Fig. 9**). Here, each message m_{2i} is said to as *properly causally precede* m_2 ($i = 1, \ldots, l$) since there is no message that p_t sends after receiving m_{2i} sending before m_2. Let $d_t(m)$ be a set of messages that properly causally precede a message m in a process p_t. $d_t(m_2) = \{m_{21}, \ldots, m_{2l}\}$ in Fig. 9. In the multimedia group protocol, there is no causal precedence between m_2 and m_{2i} unless m_2 or m_{2j} is the top or last message of an object. Let $M_t(m)$ be a set of messages which properly causally precede m and are to be ordered in the MCO protocol. N_G and N_{OG} denote the communication overheads. The larger N_G and N_{OG} are, the longer it takes to deliver messages. Let N_G be the average number of $|d_t(m)|$ and N_{OG} be the average number of $|M_t(m)|$ for every message m. N_G and N_{OG} are measured by simulation.

We make the following assumptions regarding the evaluation:

1. There are n (> 1) processes $p_1, \ldots, p_n$.
2. Each process p_t sends one object at a time and sends a total of 1000 objects.
3. Each object is sent to all the other processes.
4. Each object is decomposed into h messages.
5. Each process sends one message every τ time units. τ is a random variable between *mint* and *maxt*. The average intermessage time $\bar{\tau}$ is $(mint + maxt)/2$.
6. It takes δ time units for a message to arrive at the destination.

Figure 10 shows the ratio of N_{OG} to N_G for the number n of the processes in the group G. $\delta/\bar{\tau} = 0.25$ shows a situation in which work-

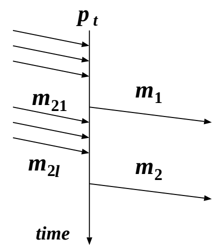


Fig. 9 Proper precedence.

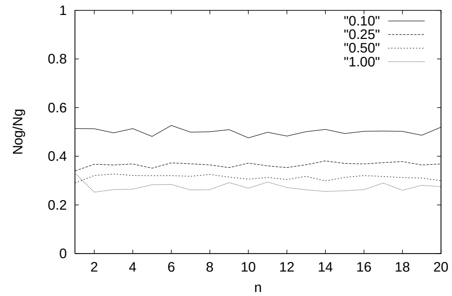


Fig. 10 Number of messages to be ordered.

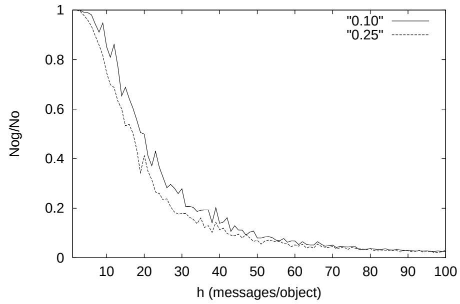


Fig. 11 Number of messages to be ordered.

stations are interconnected in a local area network. The larger $\delta/\bar{\tau}$ is, the more distant a pair of processes are. Here, each object is transmitted by twenty messages ($h = 20$). The ratio N_{OG}/N_O is almost independent of the size n of the group. For example, N_{OG}/N_O is about 0.35; that is, only 35% of the messages are handled to be causally ordered in the MCO protocol for $\delta/\bar{\tau} = 0.25$. $\delta/\bar{\tau} = 0.10$ indicates a wide area network in which about 55% of the messages are ordered in the MCO protocol.

N_{OG}/N_G shows how much the multimedia group protocol can reduce the computation and communication overheads. **Figure 11** shows N_{OG}/N_G for the number h of messages of an object where $\delta/\bar{\tau} = 0.25$, $\delta/\bar{\tau} = 0.1$, and $n = 10$. h denotes the size of each object. The larger an object, the lower the ratio of the number of messages that are causally preceded in the MCO protocol to the number in the traditional one.

6. Concluding Remarks

This paper has discussed a group protocol named the MCO protocol in which multiple processes exchange multimedia objects in a group of the processes. We defined novel types of causally precedent relations among multimedia objects, i.e. top ($\rightharpoonup$), tail ($\rightharpoondown$), partially ($\rightarrow$), and fully ($\Rightarrow$) precedent relations. We also designed the protocol to support the ordered delivery of objects in the types of the causalities. The MCO protocol is now being implemented on Unix. We discussed how the multimedia group protocol can reduce the number of network-level messages to be causally preceded through simulation. We are now extending the MCO protocol so as to satisfy the precedence relation among component objects.

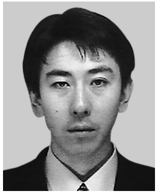
Acknowledgments This research is partially supported by the Research Institute for Technology of Tokyo Denki University.

References

1) Adelstein, F. and Singhal, M.: Real-Time Causal Message Ordering in Multimedia Systems, *Proc. IEEE ICDCS-15*, pp.36–43 (1995).
2) Baldoni, R., Mostefaoui, A. and Raynal, M.: Efficient Causally Ordered Communications for Multimedia Real-Time Applications, *Proc. IEEE HPDC-4*, pp.140–147 (1995).
3) Birman, K.: Lightweight Causal and Atomic Group Multicast, *ACM Trans. Comput. Syst.*, pp.272–290 (1991).
4) Postel, J.: User Datagram Protocol, ISI RFC768 (1980).
5) Kanezuka, T., Higaki, H., Takizawa, M. and Katsumoto, M.: QoS Oriented Flexible Distributed Systems for Multimedia Applications, *Proc. 13th Int'l Conf. on Information Networking* (*ICOIN-13*), pp.7C-4 (1999).
6) Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System, *Comm. ACM*, Vol.21, No.7, pp.558–565 (1978).
7) MPEG Requirements Group: MPEG-4 Requirements, ISO/IEC JTC1/SC29/WG11 N2321 (1998).
8) 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).
9) 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).
10) Nakamura, A. and Takizawa, M.: Reliable Broadcast Protocol for Selectively Ordering PDUs, *Proc. IEEE ICDCS-11*, pp.239–246 (1991).
11) Nakamura, A. and Takizawa, M.: Priority-Based Total and Semi-total Ordering Broadcast Protocols, *Proc. IEEE ICDCS-12*, pp.178–185 (1992).
12) Nakamura, A. and Takizawa, M.: Causally Ordering Broadcast Protocol, *Proc. IEEE ICDCS-14*, pp.48–55 (1994).
13) Prakash, R., Raynal, M. and Singhal, M.: An Adaptive Causal Ordering Algorithm Suited to Mobile Computing Environments, *Proc. J. Parallel and Distributed Computing*, Vol.41, pp.190–204 (1997).
14) Shimamura, K., Tanaka, K. and Takizawa, M.: Group Protocol for Exchanging Multimedia Objects in a Group, *Proc. IEEE IWGCC*, pp.33–40 (2000).
15) Tachikawa, T. and Takizawa, M.: Communication Protocol for Wide-Area Groups, *Proc. Int'l Computer Symp.*, pp.158–165 (1996).
16) Tachikawa, T., Higaki, H. and Takizawa, M.: Group Communication Protocol for Realtime Applications, *Proc. IEEE ICDCS-18*, pp.158–165 (1998).
17) Tachikawa, T. and Takizawa, M.: Multimedia Intra-group Communication Protocol, *Proc. IEEE HPDC-4*, pp.180–187 (1995).
18) Yavatkar, R.: MCP: A Protocol for Coordination and Temporal Synchronization in Multimedia Collaborative Applications, *Proc. IEEE ICDCS-12*, pp.606–613 (1992).

(Received May 18, 2000)
(Accepted December 1, 2000)

Kenichi Shimamura was born in 1978. He received his B.E. degrees in computers and systems engineering from Tokyo Denki Univ., Japan in 2000. He is now a graduate student of the master course in the Dept. of Computers and Systems Engineering, Tokyo Denki Univ. His research interests include distributed multimedia networks. He is a member of IPSJ.



Katsuya Tanaka was born in 1971. He received his B.E. and M.E. degrees in Computers and Systems Engineering from Tokyo Denki University, Japan in 1995 and 1997, respectively. From 1997 to 1999, he worked for NTT Data Corporation. Currently, he is a assistant in the Department of Computers and Systems Engineering, Tokyo Denki University. He received the D.E. degree from Dept. of Computers and Systems Engineering, Tokyo Denki University, Japan, in 2000. His research interest includes distributed systems, transaction management, recovery protocols, and computer network protocols. He is a member of IEEE CS and IPSJ.

Makoto Takizawa was born in 1950. He received his B.E. and M.E. degrees in Applied Physics from Tohoku Univ., Japan, in 1973 and 1975, respectively. He received his D.E. in Computer Science from Tohoku Univ. 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 Dept. of Computers and Systems Engineering, Tokyo Denki Univ. 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 univ., England since 1990. He was a program co-char of IEEE ICDCS-18, 1998 and serves on the program committees of many international conferences. He chaired SIGDPS of IPSJ from 1997 to 1999. He is IPSJ fellow. His research interest includes communication protocols, group communication, distributed database systems, transaction management, and security. He is a member of IEEE, ACM, and IPSJ.