Causally Ordered Delivery of Multimedia Objects

Kenichi Shimamura, Katsuya Tanaka, and Makoto Takizawa
Tokyo Denki University
E-mail {ken, katsu, taki}@takilab.k.dendai.ac.jp

In distributed applications like teleconferences and teleclassrooms, a group of multiple processes are cooperating, where messages exchanged among the processes are required to be causally delivered. The processes are exchanging kinds of multimedia objects in addition to traditional text data. The multimedia messages are longer than traditional messages and are structured. In this paper, we discuss new types of causally precedent relations among multimedia objects transmitted in the network. We discuss a protocol to causally deliver multimedia objects in a group of multiple processes. We also show the evaluation of the protocol.

1 Introduction

In distributed applications like teleconferences, a group of multiple processes are cooperating. Various kinds of group protocols [3, 12] are discussed so far. In the group communication, a *group* is first established among multiple processes and then messages sent by the processes are *causally*, *totally* delivered to the destination processes in the group [3, 5]. A message m_1 *causally precedes* another message m_2 if a sending event of m_1 *happens before* a sending event of m_2. In the totally ordered delivery, even messages not to be causally ordered are delivered to every common destination of the messages in a same order. In the protocols, messages transmitted at the network level are ordered independently of what kinds of information are included in the messages.

In distributed applications, various kinds of multimedia objects like image and video are exchanged among multiple processes in the group. Thus, multimedia objects are structured and are larger and more complex than the traditional data messages. In addition to causally delivering objects, a multimedia object received has to satisfy quality of service (QoS) like frame rate and number of colors required by the destination processes. For example, a destination process is allowed to receive only some part of an object as far as the part satisfies QoS requirement of the application. Some objects may have to be delivered to the applications in predetermined time units after the objects are sent. The papers [1, 2, 15] discuss the Δ-causality where Δ is the maximum delay time in the system. Tachikawa and Takizawa [13] define the Δ-ϵ causality among messages where Δ_{st} is the maximum delay time and ϵ_{st} is the maximum ratio of messages between every pair of processes p_s and p_t.

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 decomposed from o_1 and o_2 are causally delivered in every common destination process p_3 of o_1 and o_2 according to the traditional group protocols [3]. In an application, the messages of o_1 can be delivered independently of o_2. The object o_1 is also manipulated independently of o_2. In another application, the top message of o_1 is required to be delivered before the top of o_2 while the other messages can be delivered in any order. Thus, we define new types of precedent relations of messages based on the object concept. According to the precedent relations, the destination process delivers messages of objects to the application. A pair of messages not to be ordered in the precedent relations can be delivered in any order even if one of them causally precedes the other according to the traditional network-level destination. We discuss a protocol which supports the types of causally precedent relations, named *causally ordered multimedia* (*COM*) group protocol, where a fewer number of messages are causally ordered than the traditional network-level group protocols.

In section 2, we present a system model and multimedia objects. In section 3, types of causally precedent relations among multimedia objects are discussed. In section 4, we present the COM protocol for exchanging multimedia objects in a group of processes. In section 5, we show the evaluation of the COM protocol.

2 System Model

Distributed applications are realized by cooperation of a group of application processes A_1, ..., A_n ($n \geq 1$). Application processes exchange objects including multimedia data with the other processes in the group by using the network.

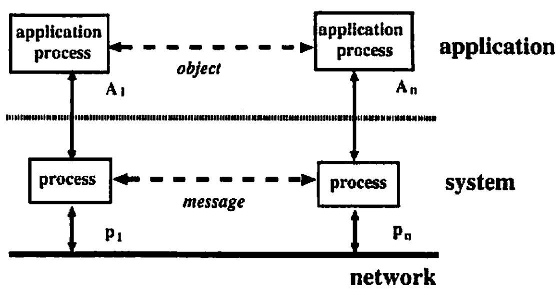

Figure 1: Layers.

An application process A_t is supported by a system process p_t (t = 1, ..., n) as shown in Figure 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. From here, let a term *process* mean a system process.

A data unit exchanged by the processes in the network is referred to as *message*. We assume that the network supports processes with synchronous communication. That is, messages are not lost and delay time between a pair of processes is bounded in the network.

An object is decomposed into a sequence of messages by a source process and the messages are delivered to the destination processes. A destination process p_t assembles received messages 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 by taking usage of the network service. We discuss a group protocol for delivering multimedia objects to processes in a group.

3 Causality of Objects

3.1 Causality of messages

Let $s_t(m)$ and $r_u(m)$ denote sending and receipt events that processes p_t and p_u send and receive a message m, respectively. By using the *happen-before* relation ($\prec$), the causally precedent relation among messages is defined as follows:

- A message m_1 *causally precedes* another message m_2 iff $s_t(m_1)$ happens before ($\prec$) $s_u(m_2)$.

Suppose three processes p_s, p_t, and p_u are exchanging messages. The process p_s sends a message m_1 to a pair of processes p_t and p_u. The process p_t sends a message m_2 to p_s and p_u after receiving m_1. Since $s_t(m_1) \prec s_u(m_2)$, m_1 *causally precedes* m_2. The process p_u has to deliver m_1 before m_2. In order to causally order the messages, the *vector clock* [5] is widely used in the group protocols [3]. Suppose there are $n(>1)$ processes $p_1, ..., p_n$ in a group G. Each process p_t manipulates a vector clock $V = \langle V_1, ..., V_n \rangle$ where each element V_u is initially 0 for $u = 1, ..., n$. When p_t sends a message m, $V_t := V_t+1$ and m carries the vector clock $m.V(= V)$ of p_t. On receipt of a message m, $V_u := \max(V_u, m.V_u)$ for $u = 1, ..., n$ in each destination process of m. For pair of vectors $A = \langle A_1, ..., A_n \rangle$ and $B = \langle B_1, ..., B_n \rangle$, $A < B$ iff $A_j \neq B_j$ for $i = 1, ..., n$ and $A_j < B_j$ for some 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 $m_1.V < m_2.V$.

The traditional group protocols [3, 5–9] discuss how to causally and totally deliver *network-level* messages, independently of what kinds of application data are carried by the messages.

3.2 Causality of objects

We discuss how a process sends and receives multimedia objects in a group G of multiple processes $p_1, ..., p_n$ $(n > 1)$. Suppose a process p_s sends an object o to another process p_t. It takes a longer time to send and receive the multimedia object since the multimedia object is larger and more complex than a traditional message. 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. Figure 2 shows three processes p_s, p_t, and p_u exchanging objects o_1 and o_2. In Figure 2(3), the process p_t starts sending messages of an object o_2 after receiving all the messages of another object o_1. According to the traditional causality theory [5], o_1 causally precedes o_2. In Figure 2(1), p_t starts sending 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 Figure 2(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.

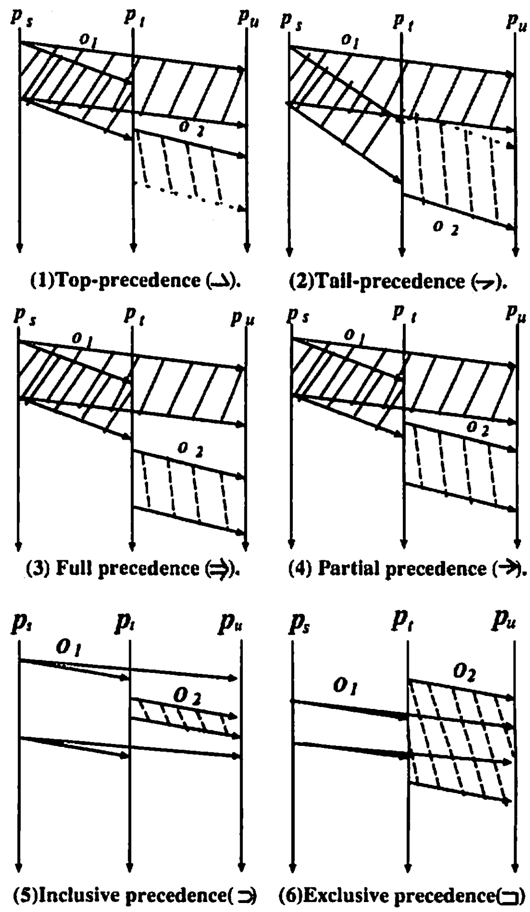

Figure 2: Precedency of objects.

[Example 1] Let us consider an example of a teleconference where the participants are distributed on 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, as shown in Figure 2. Each process supports a remote conference site where remote participants join the conference and exchanges messages with the other processes. Participants in the conference share a virtual *conference* space C which is composed of three subspaces C_s, C_t, and C_u. Each subspace shows participants attending the conference at each site. The virtual space C is displayed at each site. Each site S_i distributes its subspace object C_i including the image of the site, voice of participants, and manuscripts to be handed out to all the processes in the group (i =

s, t, u). Suppose some participant at the site S_s supported by the process p_s expresses some opinion which is shown by a voice and image object o_1. The process p_s distributes messages of the object o_1. After listening to the participant of p_s, another participant of p_t expresses a counter opinion for 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 sending o_2 after receiving all the messages of o_2. Hence, the process p_u has to receive o_2 after o_1 as shown in Figure 2(2).

Next, suppose some participant supported by the process p_s is expressing the opinion which is shown by an object o_1. While listening to the participant of p_s, another participant of p_t is leaving the conference. This image object o_2 showing his leaving the conference is distributed to the processes in the group. The process p_u has to start delivering o_2 after starting delivering o_1 as shown in Figure 2(1).

Suppose the process p_s is distributing a music object o_1 which shows that the conference will be closed soon. The music is stopped being played only after every participant leaves the conference. The process p_t is sending an object showing the participants. Hence, the process p_u has to deliver o_2 before finishing delivering o_1 as shown in Figure 2(2).□

As presented here, a pair of objects o_1 and o_2 are interrelated with respect to when processes p_s and p_t start and finish sending objects as shown in Figures 2. We discuss how a pair of objects o_1 and o_2 can be causally ordered. Let $ss_t(o)$ and $es_t(o)$ denote events that p_t starts sending and finishes sending an object o, respectively. In fact, $ss_t(o)$ and $es_t(o)$ show events that the top message and the last message of the object o are sent by p_t, respectively. $sr_t(o)$ and $er_t(o)$ also mean the receipt events of the top and last messages of the object o, respectively. Let $sr_t(o)$ and $er_t(o)$ denote events that p_t starts and finishes receiving the object o, respectively.

Suppose a precess p_t receives an object o_1 and sends another object o_2. An object o_1 is *interleaved* with another object o_2 if $sr_t(o_1) \prec ss_t(o_2) \prec er_t(o_1)$ or $sr_t(o_2) \prec er_t(o_1) \prec es_t(o_2)$ in a source process p_t of o_2. Here, the process p_t is receiving messages of the object o_1 while sending messages of o_2. Next, suppose p_t sends o_1 and o_2. o_1 is interleaved with o_2 if $ss_t(o_1) \prec ss_t(o_2) \prec es_t(o_1)$ or $ss_t(o_2) \prec ss_t(o_1) \prec es_t(o_2)$.

We now define new types of precedent relations among objects as follows:

[Definition] Let o_1 and o_2 be a pair of objects o_1 and o_2 sent by processes p_s and p_t, respectively:

1 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 $p_s = p_t$.

2 o_1 *tail-precedes* o_2 ($o_1 \rightharpoondown o_2$) iff
 - $er_t(o_1) \prec es_t(o_2)$ if $p_s \neq p_t$.
 - $es_s(o_1) \prec es_t(o_2)$ if $p_s = p_t$.

3 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 $p_s = p_t$.

4 o_1 *inclusively precedes* o_2 ($o_1 \supset o_2$) iff $o_1 \rightharpoonup o_2$ and $o_1 \rightharpoondown o_2$

5 o_1 *exclusively precedes* o_2 ($o_1 \sqsupset o_2$) iff $o_1 \rightharpoondown o_2$ and $o_2 \rightharpoonup o_1$

6 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.□

The top, tail, fully, partially, inclusively, and exclusively precedent relations are referred to as *object-causally precedent* (*o-precedent*) relations. Here, $o_1 \rightsquigarrow o_2$ shows that o_1 object-causally precedes o_2, i.e. $\rightsquigarrow \in \{\rightharpoonup, \rightharpoondown, \Rightarrow, \rightarrow, \supset, \sqsupset\}$. The process p_u is required to deliver messages of objects o_1 and o_2 so as to satisfy the o-precedent relation $\rightsquigarrow$ between o_1 and o_2.

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

[Properties] Let o_1, o_2, and o_3 be objects.

P1: $o_1 \Rightarrow o_2$ if $o_1 \Rightarrow o_3$ and $o_3 \Rightarrow o_2$.
P2: $o_1 \rightharpoonup o_2$ if $o_1 \rightharpoonup o_3$ and $o_3 \rightharpoonup o_2$.
P3: $o_1 \rightharpoondown o_2$ if $o_1 \rightharpoondown o_3$ and $o_3 \rightharpoondown o_2$.
P4: $o_1 \Rightarrow o_3$ if $o_1 \Rightarrow o_3$ and $o_3 \rightharpoonup o_2$.
P5: $o_1 \Rightarrow o_2$ if $o_1 \rightharpoondown o_3$ and $o_3 \Rightarrow o_2$.
P6: $o_1 \rightharpoonup o_2$ and $o_1 \rightharpoondown o_2$ if $o_1 \Rightarrow o_2$.
P7: $o_1 \Rightarrow o_2$ if $o_1 \rightarrow o_2$.
P8: $o_1 \rightharpoonup o_2$ and $o_1 \rightharpoondown o_2$ if $o_1 \rightarrow o_2$.
P9: $o_1 \supset o_2$ if $o_1 \supset o_3$ and $o_3 \supset o_2$.
P10: $o_1 \sqsupset o_2$ if $o_1 \sqsupset o_3$ and $o_3 \sqsupset o_2$.
P11: $o_1 \rightharpoonup o_2$ if $o_1 \supset o_2$.
P12: $o_1 \rightharpoondown o_2$ if $o_1 \sqsupset o_2$.
P13: $o_1 \Rightarrow o_2$ if $o_1 \sqsupset o_3$ and $o_3 \Rightarrow o_2$.
P14: $o_1 \rightharpoondown o_2$ if $o_1 \sqsupset o_3$ and $o_3 \rightarrow o_2$.
P15: $o_1 \rightharpoonup o_2$ if $o_1 \supset o_3$ and $o_3 \rightharpoonup o_2$.
P16: $o_1 \rightarrow o_2$ if $o_1 \rightarrow o_3$ and $o_3 \rightarrow o_2$.□

The o-precedent relations $\Rightarrow$, $\rightharpoonup$, $\rightharpoondown$, $\rightarrow$, $\supset$, and $\sqsupset$ are transitive according to the properties **P**1, **P**2, **P**3, **P**9, **P**10, and **P**16.

4 COM Protocol

We present a *causally ordered multimedia* (*COM*) protocol for supporting the object-causally ordered (OCO) delivery of multimedia objects for a group G of multiple processes p_1, ..., p_n ($n > 1$).

4.1 Object transmission

A message m sent by a process p_s carries a sequence number *seq*. *seq* is incremented by one each time p_s sends a message. Here, the processes can simultaneously send multiple objects. Two types of vectors of variables $V = \langle V_1, ..., V_n \rangle$ and $A = [A_1, ..., A_n]$ are manipulated for each process in the group G. The vectors V and A are manipulated in a way similar to the vector clock [5]. Each pair of elements V_t and A_t are used to show the number of sending events and the number of sending and receipt events occurring in p_t, respectively ($t = 1, ..., n$).

Initially, $V = \langle 0, ..., 0 \rangle$ and $A = [0, ..., 0]$. The vectors V and A are manipulated in every process p_t. First, suppose a process p_t starts sending an object o. Here, the t-th elements V_t and A_t of the vectors V and A are incremented by one:

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

The process p_t eventually finishes sending the object o. Only the variable A_t is incremented by one when p_t finishes sending an object o. However, V_t is not changed.

- $A_t := A_t + 1$;

Thus, V_t shows how many sending events of objects occur in p_t. A_t shows how many sending and receiving events occur in p_t. Here, let $o.SA$ and $o.SV$ show values of the vectors A and V, respectively, when p_t starts sending an object o. Let $o.EV$ and $o.EA$ show the values of the vectors V and A, respectively, when p_t finishes sending the object o. Hence, let $o.V$ and $o.A$ indicate the values of the vector V and A of the object o, respectively. While p_t is sending the object o, $o.V$ and $o.A$ are changed if p_t starts and finishes sending other objects. The object o carries the vector information $o.V$ and $o.A$ to the destination processes. If each message of the object o carries the current values of $o_1.V$ and $o_1.A$, the communication overheads are increased. In order to reduce the communication overheads, the values $o.SV$ and $o.SA$ are carried by a top message of the object o. That is, $m.V = o.SV$ and $m.A = o.SA$. Every message m following the top message is considered to carry $m.V = o.SV$ and $m.A = o.SA$. The value $o.EA$ is carried by a last message of the object o. Some messages may be lost due to unexpected delay of the network. In order to increase the reliability, $o.SV$ and $o.SA$ can be carried by multiple messages, e.g. the top message and one message after the top. $o.EV$ and $o.DA$ can also be carried by multiple messages.

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

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

On receiving a last message of the object o, the variables are changed as follows:

- $A_s := \max(A_s, o.EA_s)$ $(s = 1, ..., n, s \neq t)$;

The following properties among the *object-causally precedent* relations and the vectors hold:

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

- $o_1 \Rightarrow o_2$ iff $o_1.EA_v \leq o_2.SA_v (v=1, ..., n, v \neq s)$.
- $o_1 \rightharpoonup o_2$ iff $o_1.SV_v \leq o_2.SV_v (v=1, ..., n, v \neq s)$.
- $o_1 \rightharpoondown o_2$ iff $o_1.EA_v \leq o_2.EA_v (v=1, ..., n, v \neq s)$.
- $o_1 \nrightarrow o_2$ iff $o_1.EA_v \geq o_2.SA_v$, $o_1.EA_v < o_2.EA_v$, and $o_1.SV_v \leq o_2.SV_v (v=1, ..., n, v \neq s)$.
- $o_1 \supset o_2$ iff $o_1.SV_v \leq o_2.SV_v$ and $o_2.EA_s \leq o_1.EA_s$ $(v=1, ..., n, v \neq s)$.
- $o_1 \sqsupset o_2$ iff $o_1.EA_v \leq o_2.EA_v$ and $o_1.SV_s > o_2.SV_s$. $(v=1, ..., n, v \neq s)$.□

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

[Example 2] Figure 3 shows three processes p_s, p_t, and p_u exchanging objects o_1, o_2, and o_3. First, the process p_s starts sending 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 the object 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 sending 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 $o_1.SV < o_3.SV$, $o_1 \rightharpoonup o_3$.□

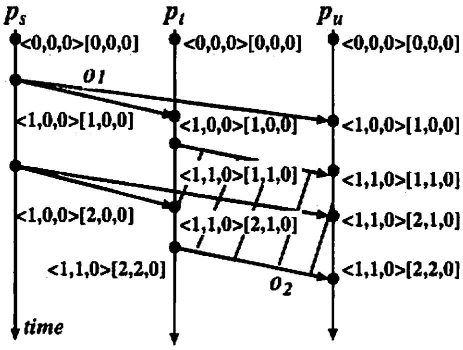

Figure 3: Object-causally ordered delivery.

4.2 Fine synchronization

In Figure 3, the object o_1 top-precedes o_2 ($o_1 \rightharpoonup o_2$). The process p_u delivers the messages of o_2 after starting delivering o_1. The process p_u can deliver the messages of o_2 just after delivering a top message of o_1. The process p_u can also deliver the object o_2 just before p_u delivers a last message of o_1. Thus, the object-causally precedent relation does not define how different it is between the starting times when the objects o_1 and o_2 are started to be transmitted. A special type of message named a *synchronization* message is sent for each object o to relate o_1 and o_2 at a smaller granularity level. Each time a synchronization message m of the object o is sent, V_t and A_t are incremented by one. The top and last messages are also synchronization ones. A sequence of messages of the object o is divided to subsequences. Each subsequence of messages starts at a synchronization message and ends at a next synchronization one. The subsequence is referred to as *segment* s_1 of the object o. The object o is considered to be a sequence of segments. In Figure 4, an object o_s is decomposed into seven messages. The messages 1, 4, and 7 are synchronization messages of the object o_s. The first segment of o_s is a sequence of messages 1,2, and 3. The second segment s_2 is a sequence of messages 4, 5, 6, and 7. The object o_s is a sequence of the segments s_1 and s_2. Here, suppose that the process p_t starts sending messages of an object o_t after p_t starts receiving the object o_s. If p_t starts sending o_t before the synchronization message 4 of o_s, every common destination p_u of o_s and o_t starts delivering o_t before the synchronization message 4 of o_s. If p_t starts sending o_t after receiving the synchronization message 4, p_u starts delivering o_t after receiving the first segment s_1 of o_s. By using the synchronization messages, segments of objects can be causally ordered.

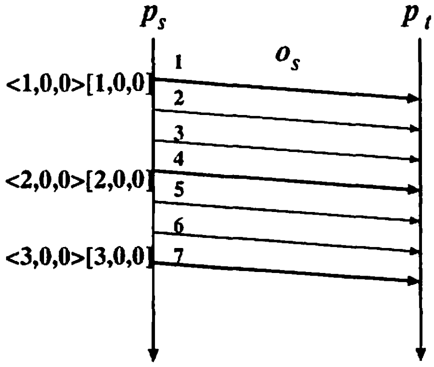

Figure 4: Synchronization messages.

Each segment s of an object o is a sequence of messages. Here, let $g(s)$ shows the number of messages included in the segment s. Let us consider a pair of objects o_1 and o_2 where $o_1 \rightsquigarrow o_2$ and o_2 is sent by a process p_t while o_1 is received by p_t, i.e. interleaving. A segment s_1 of o_1 is referred to as *directly precedes* a segment s_2 of o_2 iff the top synchronization message of s_2 is sent after receiving the top synchronization message of s_1. p_t sends a synchronization message of o_2 if p_t receives a synchronization message of o_1.

[Definition] An object o_1 *Q-precedes* o_2 iff $o_1 \rightsquigarrow o_2$, o_1 and o_2 are interleaved, and $g(s_1)/g(s_2)$ = Q for every pair of segments s_1 and s_2 where s_1 directly precedes s_2.□

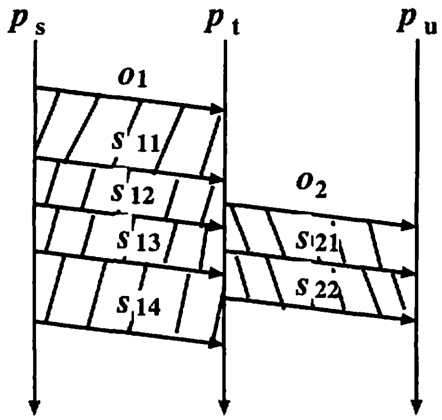

Figure 5: Quality precedence.

5 Evaluation

We evaluate the causally ordered multimedia (COM) group protocol in terms of number of network-level messages to be causally ordered compared with the traditional network-level causality. Suppose that a process p_t receives messages m_{21}, ..., m_{2l} after sending m_1 and before sending m_2 [Figure 6]. Here, each message m_{2i} is referred to as *properly causally precede* the message m_2 (i=1, ..., l) since there is no message which the process p_t sends after receiving m_{2i} before sending m_2. Let $D_t(m)$ be a set of messages which properly causally precede a message m in a process p_t. $D_t(m_2) = \{m_{21}, ..., m_{2l}\}$ in Figure 6. In the COM protocol, the top and last messages of each object carry the vectors V and A. There is no causal precedency between a pair ob messages m_2 and m_{2i} unless m_2 or m_{2j} is the top or last message of an object. Let $O_t(m)$ be a set of messages which properly causally precede m and are to be ordered in the COM protocol. Let N_G be the average number of $|D_t(m)|$ and N_{COM} be the average number of $|O_t(m)|$ for every message m. N_G and N_{COM} are considered to be metrics to evaluate the protocols because N_G and N_{COM} show numbers of messages to be compared with each message in order to causally order messages in the traditional protocol and the COM protocol, respectively. The larger N_G and N_{COM} are, the longer it takes to deliver messages. N_G and N_{COM} are measured through the simulation.

We make the following assumptions:

1. There are n (>1) processes p_1, ..., p_n in a group G.
2. Each process p_t sends one object at a time and sends totally 1000 objects.
3. A process sends an object to all the other processes in the group G.
4. Each object is decomposed into one segment composed of h ($\geq$1) messages. Each message carries data of only one object.
5. Each process sends one message every τ time units. τ is a random variable between $mint$ and $maxt$. $\bar{\tau}$ is $(mint + maxt)/2$.
6. It takes δ time units for a message to arrive at the destination.

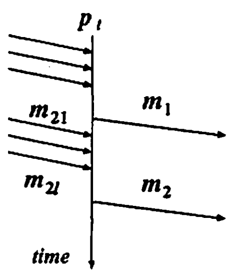

Figure 6: Proper precedence.

Figure 7 shows the ratio of N_{COM} to N_G for number n of the processes in the group G. The ratio N_{COM}/N_G shows how much the COM protocol can reduce the computation and communication overheads. The larger $\delta/\bar{\tau}$ is, the more distant a pair of processes are. $\delta/\bar{\tau}$ = 0.25 shows a situation when workstations are interconnected in a local area network. The processes exchange objects by using the local area network. Here, each object is transmitted by twenty messages (h = 20). The ratio N_{COM}/N_O is almost independent of the size n of the group. For example, N_{COM}/N_O is about 0.35, i.e. only 35% of the messages received are handled to be causally ordered in the COM protocol for $\delta/\bar{\tau}$ = 0.25, $\delta/\bar{\tau}$ = 0.1 shows a wide area network. Here, about half% of the messages which are causally ordered in the COM protocol.

Figure 8 shows the ratio. N_{COM}/N_G for number h of messages of an object where $\delta/\bar{\tau}$ = 0.25,

$\delta/\bar{\tau} = 0.1$, and $n = 10$. h shows the size of each object. The larger an object is, the less ratio of messages are causally preceded in the COM protocol than the traditional one.

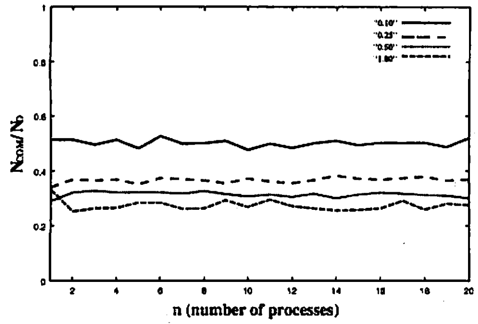

Figure 7: Evaluation.

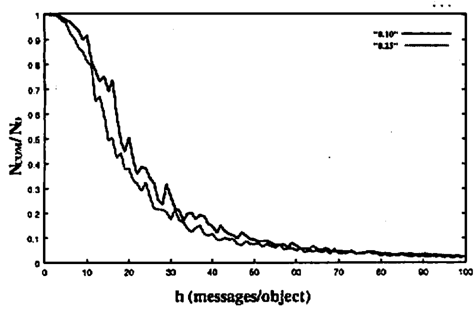

Figure 8: Evaluation.

6 Concluding Remarks

This paper discussed a group protocol named COM protocol where 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 ($\rightarrow$), fully ($\Rightarrow$), inclusive ($\supset$), and exclusive ($\sqsupset$) precedent relations. We also designed the COM protocol to support the ordered delivery of objects in the types of the causalities, which uses two types of vector clocks. We showed how the COM protocol can reduce the number of network-level messages to be causally preceded. The COM protocol is now being implemented as processes of Unix operating system by using Sun workstations.

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] 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.

[3] Birman, K., "Lightweight Causal and Atomic Group Multicast," *ACM Trans. on Computer Systems,* 1991, pp.272–290.

[4] Kanezuka, T., Higaki, H., Takizawa, M., and Katsumoto, M., "QoS Oriented Flexible Distributed Systems for Multimedia Applications," *Proc. of the 13th Int'l Conf. on Information Networking* (*ICOIN-13*), 1999, pp. 7C-4.

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

[6] 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.

[7] Nakamura, A. and Takizawa, M., "Reliable Broadcast Protocol for Selectively Ordering PDUs," *Proc. of IEEE ICDCS-11*, 1991, pp.239–246.

[8] Nakamura, A. and Takizawa, M., "Priority-Based Total and Semi-Total Ordering Broadcast Protocols," *Proc. of IEEE ICDCS-12*, 1992, pp.178–185.

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

[10] Owen, C.B. and Makedon, F., "Computed Synchronization for Multimedia Applications," Kluwer Academic, 1999.

[11] Shimamura, K., Tanaka, K., and Takizawa, M., "Group Protocol for Exchanging Multimedia Objects in a Group," *Proc. of ICDCS Int'l Workshop on Group Communications and Computations* (*IWGCC*), 2000, C33–C40.

[12] Tachikawa, T. and Takizawa, M., "Communication Protocol for Wide-area Group," *Proc. of Int'l Computer Symp.*, 1996, pp.158–165.

[13] Tachikawa, T., Higaki, H., and Takizawa, M., "Group Communication Protocol for Realtime Applications," *Proc. of IEEE ICDCS-18*, 1998, pp.158–165.

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

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