2004－DPS－118 (5)
2004／6／3

Atomicity and Causality in Real-Time Multimedia Communication

Satoshi Itaya, Tomoya Enokido, and Makoto Takizawa

Dept. of Computers and Systems Engineering
Tokyo Denki University
{*itaya, eno, taki*}*@takilab.k.dendai.ac.jp*

Abstract

A large-scale network is composed of various types of communication channels. Here, each communication channel supports Quality of Service (QoS) which may be different from others. In group communications, each process sends a message to multiple processes while receiving messages from multiple processes. In addition, messages are required to be causally delivered. Even if a process supporting enough QoS receives messages, the process cannot deliver the messages until another process supporting lower QoS receives the messages. Thus, multimedia data cannot be delivered to processes so as to satisfy the real-time constraint if a slower process is included in a group. In this paper, we discuss group communication protocols by which multimedia messages can be delivered to a process with some time constraint properties.

リアルタイム通信におけるメッセージの原子性および因果順序配送

板谷 智史　榎戸 智也　滝沢 誠
東京電機大学大学院 理工学研究科 情報システム工学専攻
E-mail {itaya, eno, taki}@takilab.k.dendai.ac.jp

大規模ネットワークでは複数のコンピュータがQoSの異なる通信チャネルで相互接続された分散型システムとなっている。テレカンファレンスシステムや遠隔授業といった分散マルチメディアアプリケーションは、複数のプロセスが映像・音声などのマルチメディアデータを送受信し、協調動作を行うことによって実現されている。大規模ネットワークでは、各チャネルが支援するQoSが異なるため十分なグループ通信サービスを提供することができない。本論文では各プロセス・ネットワークの同期性、QoS情報の特性に着目したグループ通信プロトコルの研究を行う。

1. Introduction

In distributed applications, multimedia data is exchanged among processes in networks like Gigabit and ATM networks [1]. Multimedia communication protocols like RTP and RSVP are developed so far, by which a large volume of multimedia data can be efficiently transmitted to one or more than one process. One-to-one and one-to-many communication protocols to satisfy Quality of Service (QoS) requirement like delay time, bandwidth, and message loss ratio are discussed in papers [1, 8].

In distributed applications, a collection of multiple processes are cooperating by exchanging messages. A *group* is first established among multiple processes and then messages are exchanged among the processes. Here, messages have to be *causally delivered* to multiple destination processes in the group [2]. A message m_1 *causally precedes* another message m_2 if and only if (*iff*) a sending event of m_1 *happens before* a sending event of m_2 [4]. Various types of the group communication protocols which support a group of multiple processes with the causally ordered delivery of messages have been discussed [2, 3, 6]. In a *centralized* approach to realizing group communication, there is one controller site through which processes are exchanging messages. The centralized approach is not suited to realize real-time multimedia applications including multiple processes, especially distributed in a wide-area network due to longer delay time. We take a fully *distributed* approach where every process directly sends a message to destination processes in a group of processes in order to satisfy real-time constraints of multimedia data. Each process receives messages from multiple sites. The process has to causally order messages received from multiple processes by itself, e.g. messages are ordered by using vector clock [4]. In addition, the process is required to send a message to each destination process while receiving messages from multiple processes so that QoS requirement is satisfied.

Since each communication channel between processes may support different QoS due to distance and congestions, a message sent by a process may not arrive at every destination process at a same time. Every destination process cannot deliver messages received until the slowest destination process receives the messages. Thus, time constraint is not satisfied if a group includes a less-qualified process. We discuss requirements of group communication like real-time constraint in addition to the atomic and causally ordered delivery of messages. We discuss how to synchronize transmissions of messages to multiple pro-

cesses and receipts of messages from multiple processes so as to satisfy the group communication requirements in the fully distributed group.

In section 2, we present a system model. In section 3, we discuss a model for transmitting and receiving multimedia messages in a group. In section 4, we discuss the atomicity and causality of multimedia messages in group communication.

2. System Model

2.1 Channel

Cooperation of multiple application processes $AP_1, \ldots, AP_n (n>1)$ is supported by system processes $p_1, \ldots, p_n (n>1)$. A collection of multiple peer processes is referred to as *group* G. The group communication service is provided for the application processes by multiple system processes. The network is modeled to be a collection of logical communication channels. Processes communicate with each other by taking usage of communication service supported by channels. There is a channel $C_{ij} = \langle p_i, p_j \rangle$ between every pair of processes p_i and p_j in the group G. Each channel $\langle p_i, p_j \rangle$ is *bidirectional*, i.e. $\langle p_i, p_j \rangle$ exists if $\langle p_j, p_i \rangle$ exists and is *synchronous*, i.e. the maximum delay time is bounded.

An application process AP_i sends a message m to one or more than one destination process in a group. Let $dst(m)$ denote a collection of destination processes of a message m, which is a subset of a group. Let $src(m)$ show a source process which sends a message m. A message m is transmitted from a process AP_i to every destination process AP_j in $dst(m)$ via a channel $C_{ij} = \langle p_i, p_j \rangle$. A message m sent by an application process is decomposed into a sequence $pkt(m)$ $(l \geq 1)$ of *packets* $\langle t_1, \ldots, t_l \rangle$ [Figure 1]. A packet is a unit of data transmission in a network. The process p_i transmits a packet sequence $pkt(m)$ to every destination process p_j of the message m via a channel $C_{ij} = \langle p_i, p_j \rangle$. A destination process p_j receives *packets* sent by the process p_i through the channel C_{ij} and assembles the packets into the message m. Then, the message m is delivered to the destination application process AP_j.

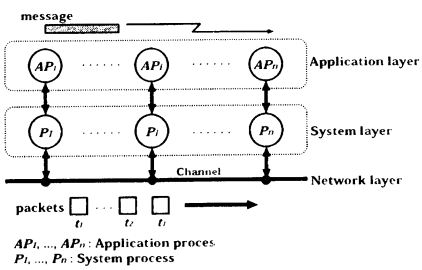

Figure 1. System model.

2.2 QoS

Each channel $\langle p_i, p_j \rangle$ supports Quality of Service (QoS), which is denoted by $Q(\langle p_i, p_j \rangle)$ or Q_{ij}. QoS is characterized by parameters; bandwidth(bw) [bps], packet loss ratio(pl) [%], and delay(dl) [msec]. Each QoS instance is a tuple of values $\langle v_1, \ldots, v_m \rangle$ where each v_i is a value of QoS parameter q_i $(i = 1, \ldots, m)$. Let $\mathbf{Q}$ be a set of QoS parameters $q_1, \ldots, q_m$. Let A and B be QoS instances $\langle a_1, \ldots, a_m \rangle$ and $\langle b_1, \ldots, b_m \rangle$, respectively. Each QoS value a_i of the parameter q_i in the QoS instance A is shown by $q_i[A]$. If a_i is better than b_i $(a_i \succ b_i)$ for every parameter q_i, A *precedes* B $(A \succ B)$. $A \succeq B$ iff $A \succ B$ or $A = B$. A pair of QoS instances A and B are *uncomparable* $(A \parallel B)$ iff neither $A \succeq B$ nor $A \preceq B$. A *preference* relation "$\rightarrow$" is a partially ordered relation on QoS parameters $q_1, \ldots, q_m$, i.e. $\rightarrow \subseteq \mathbf{Q}^2$. "$q_i \rightarrow_t q_j$" shows that a process p_t prefers a QoS parameter q_i to another parameter q_j. "$q_i \rightarrow q_j$" shows "$q_i \rightarrow_t q_j$" for some process p_t in a group. The preference relations "$\rightarrow_t$" and "$\rightarrow$" are asymmetric. For example, $bw \rightarrow_t pl$ if bandwidth (bw) is more significant than packet loss ratio (pl) for a process p_t. For every pair of QoS parameters q_i and q_j, $q_i \cup q_j$ and $q_i \cap q_j$ show least upper bound (lub) and greatest lower bound (glb) of q_i and q_j, respectively, with respect to the preference relation "$\rightarrow$". Let P_t be a partially ordered set $\langle \mathbf{Q}, \rightarrow_t \rangle$, named *preference* of a process p_t. $\langle \mathbf{Q}, \rightarrow \rangle$ is $\langle \mathbf{Q}, \bigcup_{t=1,\ldots,n} \rightarrow_t \rangle$. For example, $\mathbf{Q} = \{bw, pl, dl\}$ and $bw \rightarrow_t pl$ and $dl \rightarrow_t pl$ for a process p_t. A preference P_t is $\langle \mathbf{Q}, \{bw \rightarrow_t pl, dl \rightarrow_t pl\} \rangle$. A QoS parameter q is referred to as *maximal* in $\mathbf{Q}$ iff there is no QoS parameter q' in $\mathbf{Q}$ such that $q' \rightarrow q$.

Let A and B be QoS instances $\langle$128[Mbps], 100[msec], 0.1[%]$\rangle$ and $\langle$64[Mbps], 100[msec], 0.01[%]$\rangle$, respectively, for QoS parameters $\mathbf{Q} = \langle bw, pl, dl \rangle$. Here, 64 $\succ$ 128[Mbps], 100 = 100[msec], and 0.1 $\succ$ 0.01[%]. If the bandwidth (bw) and delay time (dl) are more significant than the packet loss ratio (pl) in an application, $bw \rightarrow pl$ and $dl \rightarrow pl$ in a preference P of the application. Here, bw and dl are maximal. The QoS instance A is *more preferable than* B with respect to the preference P $(A \succ_P B)$ since the bandwidth and delay time of A are better than B although the packet loss ratio of B is better than A. Let a_i denote a value of a QoS parameter q_i in $\mathbf{Q}$ in a QoS instance A. A preference relation "$A \succ_P B$" with respect to a preference P holds iff

1. For every maximal parameter q_i in $\mathbf{Q}$, $a_i \succeq_P b_i$.
2. If $a_i = b_i$ for every maximal parameter q_i in $\mathbf{Q}$, $A' \succeq_P B'$ for QoS instances A' and B' obtained by removing QoS instances of the maximal QoS parameters.

QoS requirement R_i of a process p_t is given a pair of parameters $MaxQ_t$ and $MinQ_t$, which show max-

imal and minimal QoS required by p_t, respectively. A process p_t is referred to as *satisfy* QoS requirement R_t if $MaxQ_t \succeq_t Q_t \succeq_t MinQ_t$ for every QoS instance Q_t taken by the process p_t. $MaxQ_t$ shows the most preferable QoS which can be realized in the implementation and $MinQ_t$ indicates the least preferable QoS of the process p_t. Suppose a process p_s sends messages to another process p_t. We discuss how much QoS instance a pair of processes p_s and p_t agree on in order to communicate with one another. Let Q be QoS instance. Q is referred to as *satisfiable* for a pair of QoS requirements R_s and R_t of processes p_s and p_t, respectively, if $MaxQ_s \succeq_s Q \succeq_s MinQ_s$ and $MaxQ_t \succeq_t Q \succeq_t MinQ_t$. Q is *maximally satisfiable* for the requirements R_s and R_t i.e. $Q = R_s \cup R_t$ iff Q is satisfiable for R_s and R_t and there is no QoS instance Q' satisfiable for R_s and R_t such that $Q' \succeq_s Q$ and $Q' \succeq_t Q$. Similarly, we define a minimally satisfiable QoS instance ($R_s \cap R_t$) for R_s and R_t.

3. Data Communication Model

3.1 Transmission

A process p_i sends packets $t_1, \ldots, t_l$ ($l \geq 1$) of a message m to every destination process p_{ij} in $dst(m)$ ($j = 1, \ldots, k_i$). There are following ways to transmit a packet sequence $pkt(m)$ ($=\langle t_1, \ldots, t_l \rangle$) to all the destination processes [Figure 2]:

1. *Synchronous* transmission: A process p_i sends each packet t_h to every destination process p_{ij} through a channel C_{ij}. Here, each packet t_h is sent in each channel C_{ij} after t_{h-1} is sent in every channel (h=1, ..., l).
2. *Partial-synchronous* transmission: A process p_i sends a number n_j of packets to a process p_{ij} while sending a number n_h of packets to another process p_{ih}. Here, the ratio $n_1 : \cdots : n_{k_i}$ is the transmission synchronization ratio of the process p_i to the processes $p_{i1}, \ldots, p_{ik_i}$.
3. *Asynchronous* transmission: A process p_i sends a sequence $pkt(m)$ of packets through each channel independently of the other channels.

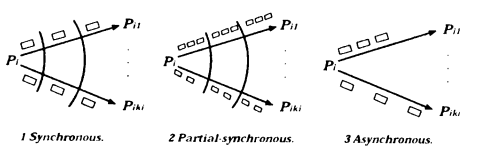

Figure 2. Types of transmission.

Ssnd(t, C) shows a procedure to send a packet t through a channel C. **Asnd**(T, C) shows a procedure to send a sequence T of packets $t_1, \ldots, t_l$ through a channel C, i.e. **for** $h = 1, \ldots, l$ **do** { **Ssnd**(t_h, C);}. A packet sequence T is decomposed into a sequence $\langle T_j^1, \ldots, T_j^{g_j} \rangle$ of segments for a process p_{ij}. A segment is a subsequence of the packets. Here, each segment is composed of the same number of packets. Let $|T_j^k|$ show the number of packets in a segment T_j^k. The ratio $|T_1^1| : \cdots : |T_{k_i}^1|$ shows the transmission synchronization ratio for the destination processes p_{i1}, ..., p_{ik_i}. **PSsnd**(T_j^k, C_{ij}) shows a procedure to send a segment T_j^k of packets in a channel C_{ij}. If the transmission synchronization ratio is $1:\cdots:1$, a process p_i sends packets at a same rate. Here, a notation $F_1 \parallel F_2$ means that a pair of procedures F_1 and F_2 are independently, possibly concurrently performed. For example, $F_1 \parallel F_2$ is realized by creating a thread for each of the procedures F_1 and F_2. The ways of transmissions can be realized by the following procedures:

1. Synchronous transmission:
   for $h = 1, \ldots, l$ **do**
   { **Ssnd**(t_h, C_i) $\parallel \ldots \parallel$ **Ssnd**(t_h, C_{ik_i}); }
2. Partial-synchronous transmission:
   for $h = 1, \ldots, g$ **do**
   {if $T_1^h \neq \phi$, **PSsnd**(T_1^h, C_{i1})} $\parallel \ldots \parallel$
   {if $T_{k_i}^h \neq \phi$, **PSsnd**($T_{k_i}^h$, C_{ik_i})};
3. Asynchronous transmission:
   Asnd(T, C_{i1}) $\parallel \ldots \parallel$ **Asnd**(T, C_{ik_i});

In the synchronous transmission, each packet is multicast. After a packet is multicast, a succeeding packet is multicast. In the asynchronous transmission, a sequence of packets are transmitted for each channel independently of the other channels. If destination processes have different maximum receipt rates, the source process p_i can take the partial-synchronous transmission. The transmission synchronization ratio shows the receipt ratio of the destination processes.

Each destination process p_{ij} of a message m sent by a process p_i has some QoS requirement R_{ij}. A process p_i has to deliver a message m to every destination process p_{ij} so as to satisfy the QoS requirement R_{ij}. Let $Q_{ij}(t_k)$ show QoS of a packet t_k transmitted in a channel $C_{ij} = \langle p_i, p_{ij} \rangle$. When a group G is established among processes, every pair of processes p_i and p_{ij} do the negotiation on the preference P_{ij} to be used when a process p_i sends messages to the process p_{ij}. Let "$\succ_{ij}$" denote a preference relation "$\succ_{P_{ij}}$". $Q_{ij}(t_k)$ is required to be satisfiable for QoS requirement R_{ij}. In fact, $Q_{ij}(t_k)$ is shown in terms of bits. There are two ways to transmit a message m to multiple processes $p_{ij}, \ldots, p_{ik_i}$[Figure 3]:

1. *Quality(Q)-balanced* transmission: For each packet t_h of a message m, $Q_{i1}(t_h) = \ldots = Q_{ik_i}(t_h)$.
2. *Quality(Q)-unbalanced* transmission: For some pair of channels C_{ij} and C_{ih}, $Q_{i1}(t_k) \neq Q_{ij}(t_k)$.

In the first way, each packet of a message m is sent with same QoS in every channel. That is, a same packet is sent in every channel. In the second way,

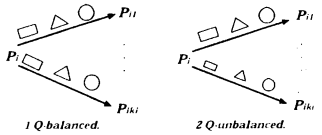

Figure 3. Quality-based transmission.

QoS in each channel is not necessarily same. For example, some channel supports lower bit-rate. In order to synchronously transmit packets to multiple destination processes, packets with smaller number of bits are transmitted.

Let us consider a synchronous transmission of a message m to multiple destination processes in $dst(m)$. If each channel supports enough QoS, a process p_i can synchronously send a same packet in every channel. Here, since each channel supports the same QoS, this is *quality(Q)-balanced* transmission. The Q-balanced, synchronous transmission is referred to as *fully synchronous*. If some channel C_{ij} does not support enough QoS due to congestions and network faults, the process p_i sends a packet t_k with less QoS in the channel C_{ij} than the others. That is, $Q_{ij}(t_k) \prec_{ij} Q_{ih}(t_h)$ for some channel C_{ik} $(h \neq k)$. Next, suppose QoS is more significant than the synchronous requirement in an application. The process p_i sends the packets in the channel C_{ij} more slowly than the other channels. That is, the process p_i asynchronously sends packets of the message m. The Q-unbalanced, asynchronous transmission is referred to as *independent*.

3.2 Receipt

A process p_i receives messages from one or more than one process in a group G. There are following ways for a process p_i to receive messages from multiple processes $p_{i1}, \ldots, p_{ik_t}$ $(k_i \geq 1)$:

1. *Synchronous* receipt: A process p_{ij} sends a sequence $pkt(m_j)$ of packets $t_{j1}, \ldots, t_{jl_i}$ $(l_i \geq 1)$ of a message m_j to a process p_i (j=1, ..., k_i). The process p_i receives a packet t_{jh} from each process p_{ij} after receiving a packet t_{fh-1} from every process p_{if} (f=1, ..., k_i).
2. *Partial-synchronous* receipt: A packet sequence $pkt(m_j)$ of a message m_j from a process p_{ij} is decomposed into a sequence $\langle S_j^1, S_j^2, \ldots, S_j^{g_j} \rangle$ of segments. Each segment includes the same number of packets, i.e. $|S_j^1| = |S_j^2| = \ldots = |S_j^{g_j - 1}| = NS_j$ and $|S_j^{g_j}| \leq NS_j$ for a process p_{ij}. The process p_i receives the hth segment S_j^h from a process p_{ij} after receiving the $(h-1)$th segment S_f^{h-1} from every process p_{if} $(f = 1, \ldots, k_i)$. Here, the ratio $NS_1 : \cdots : NS_{k_i}$ shows the receipt synchronization ratio for $p_{i1}, \ldots, p_{ik_i}$.
3. *Asynchronous* receipt: A process p_i receives packets from each process p_{ij} independently of the other processes.

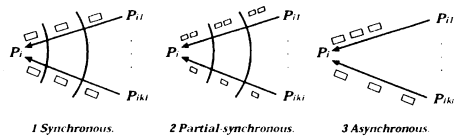

Figure 4. Types of receipt.

"t := **Srec**(C)" shows a procedure to receive one packet through a channel C and store the packet into a buffer t. "$C = \phi$" means "end of transmission". Let T be a sequence of buffers $t_1, t_2, \ldots$, for storing one message, where each buffer t_l can admit one packet. "T := **Arec**(C)" shows a procedure to receive a sequence of packets into a buffer sequence $T = \langle t_1, \ldots \rangle$ for a channel C. **Arec** is realized by the procedure: **while** $C \neq \phi$ **do** $\{t_h$:= **Srec**(C); $h := h + 1;\}$. Let T be a buffer for storing one segment. "T := **PSrec**(C)" shows a procedure to receive a segment of packets into a sequence T of buffers. The ways to receive a packet sequence $pkt(m_j)(= \langle t_{j1}, \ldots, t_{jl_j} \rangle)$ of a message m_j are realized as follows:

1. Synchronous receipt:
   while some $C_{ij} \neq \phi$ **do** {
   t_{1h} := **Srec**(C_{i1}) $\| \ldots \|$ $t_{k_i h}$:= **Srec**(C_{ik_i});
   $h := h + 1$; }
2. Partial-synchronous receipt:
   while some $C_{ij} \neq \phi$ **do** {
   T_1^h := **PSrec**(C_{i1}) $\| \ldots \|$ $T_{k_i}^h$:= **PSrec**(C_{ik_i});
   $h := h + 1$; }
3. Asynchronous receipt:
   T_1 := **Arec**(C_{i1}) $\| \ldots \|$ T_{k_i} := **Arec**(C_{ik_i});

As discussed in transmission of messages, there are following ways to receive messages from multiple processes with respect to QoS:

1. *Quality(Q)-balanced* receipt: A process p_i receives packets with same QoS from each destination process p_{ij}.
2. *Quality(Q)-unbalanced* receipt: A process p_i receives packets with different QoS from different destination processes.

If a process synchronously receives messages in a quality (Q)-balanced way, the process is referred to as *fully receive* packets. If a process asynchronously receives packets in a Q-unbalanced way, the process is referred to as *independently receive* packets.

3.3 Receipt-transmission

Suppose there are three processes p_s, p_t, and p_u exchanging messages. A process sends messages to processes while receiving messages from other processes. Suppose a process p_t sends a message m_1 while receiving a message m_2 from a process p_s. There are

following ways to send messages while receiving messages:

1. *Synchronous* receipt-transmission: A process p_t sends one packet of a message m_1 each time p_t receives one packet of m_2 from a process p_s.
2. *Partial-synchronous* receipt-transmission: A process p_t sends some number n_1 of packets of a message m_1 to a process p_u while receiving some number n_2 of packets of m_2 from p_s. The ratio n_1:n_2 shows the receipt-transmission synchronization ratio of the process p_t to p_s.
2. *Asynchronous* receipt-transmission: A process p_t sends packets of m_1 independently of receiving packets of m_2 from p_s.

In the synchronous and partial-synchronous receipt-transmission ways, every common destination process p_u of messages m_1 and m_2 is required to synchronously receive the messages m_1 and m_2 from the processes p_s and p_t, respectively. The 1:1 partial-synchronous receipt-transmission way is just a synchronous way. A process sends packets to a pair of processes p_t and p_u. The process p_t sends two packets to the process p_u while receiving three packets from the process p_s. Here, the synchronization ratio is 3:2. The process p_u is required to synchronously receive messages from the processes p_s and p_t with the receipt synchronization ratio 3:2 for the processes p_s and p_t. Here, a segment of a message m_1 includes three packets and a segment of m_2 includes two packets.

There are following types of receipt-transmission with respect to QoS:

1. *Quality(Q)-balanced* receipt-transmission: Packets received from a process p_s and packets sent by another process p_t have same QoS.
2. *Quality(Q)-unbalanced* receipt-transmission: Packets received from p_s and packets sent by p_t have different QoS.

4. Atomic and Causal Delivery

4.1. Group communication service

A group is a collection of peer processes which are cooperating to achieve some objectives. In the group communication, a process sends a message to multiple destination processes while receiving messages from multiple processes in a group. Let $s_i(m)$ be a sending event of a message m in a process p_i. The *happens before* relation on events in a distributed system is defined by Lamport [3]. A message m_1 is referred to as *causally precede* another message m_2 ($m_1 \rightarrow m_2$) iff a sending event $s_i(m_1)$ happens before a sending event $s_j(m_2)$ [3]. Every common destination process of messages m_1 and m_2 is required to receive m_1 before m_2 if $m_1 \rightarrow m_2$. Messages are causally delivered by using logical clocks like linear clock [3] and vector clock [4]. In addition, a process receiving a message m can deliver the message m only if every other destination process m surely receives the message m. In the papers [5,6], a protocol to atomically deliver messages in a group in presence of message loss is discussed.

Let us consider a group of four processes p_1, p_2, p_3, and p_4 [Figure 5]. Suppose the process p_1 sends a pair of messages m_1 and m_2 to each of the processes p_3 and p_4. The process p_2 sends a message m_3 to the processes p_3 and p_4 after receiving the message m_1 and then sends a message m_4 after receiving m_2. Here, the message m_1 causally precedes the messages m_3 and m_4 ($m_1 \rightarrow m_3$, and $m_1 \rightarrow m_4$) and the message m_2 causally precedes m_4 ($m_2 \rightarrow m_4$). Suppose the process p_3 receives m_3 and m_4 from p_2 and receives m_1 from p_1 but does not receive m_2 from p_1 due to the communication delay. The process p_3 can deliver m_1 but cannot deliver m_3 and m_4 because the message m_2 following the message m_1 from p_1 might causally precede m_3. The process p_3 has to wait for a message from the process p_1.

Next, let us consider a pair of the processes p_3 and p_4 which receive messages from p_1 and p_2. The process p_4 receives the messages m_1, m_3, m_2, and m_4 in this sequence. On receipt of the message m_2, the process p_4 can deliver the messages m_1 and m_3. However, the process p_4 cannot deliver the message m_2 because the other destination process p_3 has not received m_2 yet. The process p_4 has to wait until p_4 knows that the process p_3 receives the message m_2.

If a communication channel C_{13} between processes p_1 and p_3 implies smaller bandwidth than another channel C_{23}, the process p_3 cannot deliver messages from the process p_2. If the process p_1 sends real-time multimedia data, the process p_4 cannot satisfy the real-time requirement even if the process p_4 receives all the messages. The delivery time depends on the slowest process.

There are two types of requirement for group communication, *synchronization* and *QoS* as discussed in the preceding section. If the synchronization requirement is preferable to QoS, the quality is degraded if the synchronization requirement is not satisfied, i.e. Q-unbalanced way is taken. If the QoS requirement is preferable to the synchronization one, the synchronization constraint is weaker, i.e. partial-synchronous and asynchronous ways are taken.

4.2. Data communication instances

Let **ST** and **QT** be synchronization and QoS types of transmission and receipt, respectively, i.e. **ST** = {Synchronous (S), Partial-synchronous (PS), Asynchronous (A)} and **QT** = {Quality-balanced (QB), Quality-unbalanced (QU)}. A data communication instance is determined by a tuple $\langle s, q \rangle \in$ **ST** $\times$ **QT** which is a combination of a synchronization type $s \in$

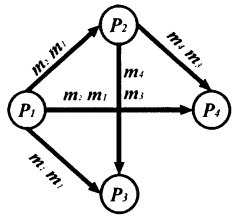

Figure 5. Causality.

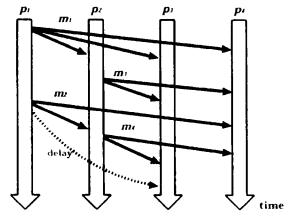

Figure 6. Causality in a group.

ST and QoS type $q \in$ **QT**. There are six data communication instances as shown in Table 1. For example, $\langle S, QB \rangle$ shows a process takes a synchronous and Q-balanced transmission.

Table 1. Types of data communication.

Instances	Types of communication
$\langle S, QB \rangle$	*⟨Synchronous, Q-balanced⟩*
$\langle S, QU \rangle$	*⟨Synchronous, Q-unbalanced⟩*
$\langle PS, QB \rangle$	*⟨Partial-synchronous, Q-balanced⟩*
$\langle PS, QU \rangle$	*⟨Partial-synchronous, Q-unbalanced⟩*
$\langle A, QB \rangle$	*⟨Asynchronous, Q-balanced⟩*
$\langle A, QU \rangle$	*⟨Asynchronous, Q-unbalanced⟩*

A process selects a data communication instance which satisfies application requirements. Then, each process sends and receives messages by using the selected data communication type. If an application requires the strict atomic receipt, a process selects an instance $\langle S, QB \rangle$ or $\langle S, QU \rangle$, i.e. each process synchronously sends and receives messages. If an application does not require strict atomic and real-time communication, a process selects an instance $\langle PS, QB \rangle$ or $\langle PS, QU \rangle$. On the other hand, if a process requires real-time communication, a process selects an instance $\langle A, QB \rangle$ or $\langle A, QU \rangle$, i.e. each process independently sends and receives messages. If quality of data is the most significant for an application, each process takes the Q-balanced way. In another case, if quality is not significant for an application, each process takes Q-unbalanced one. We consider that data communication instance which satisfies group communication service will be taken in the group.

A group G includes processes with different QoS and synchronization services. If every process in a group take a same data communication instance, the group can only support applications with the minimum service to be supported. Hence, the processes in the group G are classified into six types of subgroups. The types are ones shown in Table 1. In some type α of a subgroup G_α, every process can realize the requirement $\alpha \in \{ \langle S, QB \rangle, \langle S, QU \rangle, \langle PS, QB \rangle, \langle PS, QU \rangle, \langle A, QB \rangle, \langle A, QU \rangle \}$. Messages are transmitted by the type α data transmission procedure in each subgroup G_α.

5. Concluding Remarks

This paper discusses a group communication protocol to exchange multimedia messages among multiple processes under some synchronization and QoS requirements. Multimedia messages are exchanged among multiple processes in a group so as to satisfy QoS required by applications. We discussed how to transmit and receive messages; synchronous, partial-synchronous, asynchronous, quality-balanced, and quality-unbalanced ways. We also discussed the atomic and causally ordered delivery of messages with synchronization and QoS requirements. We are now designing the protocol for exchanging multimedia data in a group of multiple processes.

References

[1] ATM Forum. Traffic Management Specification Version4.0. 1996.

[2] K. Birman. Lightweight Causal and Atomic Group Multicast. *ACM Trans. on Computer Systems*, 9(3):272-290, 1991.

[3] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Comm. ACM*, 21(7):558-565, 1978.

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

[5] A. Nakamura and M. Takizawa. Priority-Based Total and Semi-Total Ordering Broadcast Protocols. *Proc. of the 12th IEEE ICDCS*, pages 178-185, 1992.

[6] A. Nakamura and M. Takizawa. Causally Ordering Broadcast Protocol. *Proc. of IEEE the 14th ICDCS*, pages 48-55, 1994.

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

[8] T. Tojo, T. Enokido, and M. Takizawa. Notification-Based QoS Control Protocol for Multimedia Group Communication in High-Speed Networks. *Proc. of IEEE the 24th ICDCS*, pages 644-651, 2004.