Protocol for Synchronizing Multimedia Objects Exchanged in a Group of Processes

Seiichi Hatori and Makoto Takizawa
Department of Computers and Systems Engineering
Tokyo Denki University
E-mail {hatori, taki}@takilab.k.dendai.ac.jp

In distributed applications, a group of multiple processes are cooperating by exchanging multimedia objects. If messages transmitted in a network are causally delivered by using traditional group protocols, computation and communication overheads are increased due to large size and complex structure of multimedia object. In this paper, we discuss new types of causally precedent relations among multimedia objects transmitted in a group of multiple processes. We also discuss a protocol to causally deliver multimedia objects with QoS in a group of processes.

グループ通信でのマルチメディアオブジェクトの同期プロトコル

羽鳥 精一　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

テレビ会議等の分散応用では複数のプロセスにより送信されたメッセージは因果順序に配送されなければならない。マルチメディアデータも通信されるが、従来のグループプロトコルを用いて因果順序配送を行うと、通信と処理の負荷が増大してしまう。さらに、Quality of Service(QoS) を考慮して配送する必要がある。本論文では、マルチメディアメッセージ間の因果関係を基に、QoS を考慮した因果順序配送プロトコルを提案する。

1 Introduction

In distributed applications, a group of multiple processes are cooperating. Most distributed applications like teleconferences are realized in centralized control. That is, there is one controller process which forwards messages sent by processes to destination processes. Here, it takes at least two rounds to deliver messages and the system is less reliable and available due to the fault of the centralized controller. On the other hand, each process directly exchanges messages with the other process in distributed control. In group communications [3, 9–12], messages sent by processes are *causally* delivered to destination processes in the group. The vector clocks [9] is widely used to causally order messages in distributed group communication. In distributed applications, various kinds of multimedia objects like image and video are exchanged among multiple processes in the group. An object is decomposed into a sequence of messages. If a pair of processes p_1 and p_2 send objects o_1 and o_2 to a process p_3, respectively, messages decomposed from o_1 and o_2 are causally delivered to p_3 according to the traditional group protocols. The messages of the object o_1 can be delivered independently of the object o_2 if o_1 is manipulated independently of o_2 in an application. Shimamura and Takizawa [14] define novel types of precedent relations named *Object-(O-)precedent* relation of messages based on the object concept. According to the *O*-precedent relations, the destination processes deliver messages of objects. A pair of messages not to be ordered in the *O*-precedent relations can be delivered in any order even if one of the messages causally precedes the other message according to the traditional network-level definition. In order to support QoS required by applications, message sequence of objects should be related at a smaller granularity. Shimamura and Takizawa [15] discuss how granules of objects are related in the networks and present a *causally ordered multimedia* (*COM*) group protocol which supports the types of causally precedent relations, where a fewer number of messages are causally ordered than the traditional network-level group protocols. In another way to synchronize and order multiple messages, time information attached in messages like RTP [13] can be used. However, time is not accurate in distributed systems and the causality among events should rather hold as discussed by Lamport [8]. Some communication channels may not support enough QoS due to congestions. In the example of three processes p_1, p_2, and p_3, suppose QoS supported by a channel between processes p_1 and p_3 is so much degraded that the object o_3 received by p_3 does not support enough QoS required by an application. Here, p_3 can deliver the object o_2 sent by p_2 independently of o_3. Thus, it is meaningful to make an object o_1 O-precede another object o_2 only if both of the objects o_1 and o_2 support enough QoS for applications. In this paper, we newly discuss QoS-based O-precedent relation among multimedia objects exchanged in a group of multiple processes.

In section 2, we present a system model and types of causally precedent relations among multimedia objects. In section 3, we discuss QoS-based precedent relation of objects.

2 System Model

2.1 System configuration

Distributed applications are realized by cooperation of a group of multiple application processes $A_1, ..., A_n$ (n>1). Application processes exchange objects including multimedia data with the other processes in the group by taking usage of undering networks. An application process A_t is supported by a system process p_t (t = 1, ..., n). The process p_t takes an object from the application process A_t 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 *message* is a data unit exchanged among processes. We assume the underlying network supports every pair of processes with *synchronous* communication [5], i.e. messages are not lost and maximum delay time is bounded. In our implementation, a reliable transport protocol like TCP/IP is used as the network service.

2.2 Causality

An object o is decomposed into a sequence $\langle m_1, ..., m_h \rangle$ of messages by a source process and the messages are delivered to the destination processes. Here, m_1 and m_k are referred to as *top* and *last* messages of the object o, respectively. 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 processes is coordinated by a *group protocol* which supports reliable, efficient communication service by taking usage of the network service.

Let $s_t(m)$ and $r_u(m)$ denote sending and receipt events showing that a pair of processes p_t and p_u send and receive a message m, respectively. By using the *happen-before* relation ($\prec$) among events [8], a message m_1 *causally precedes* another message m_2 iff (if and only if) $s_t(m_1) \prec s_u(m_2)$ [8]. In Figure 1, a message m_1 causally precedes another message m_2. Due to the communication delay, p_3 may receive m_3 before m_1. A process p_3 is required to deliver m_1 before m_2.

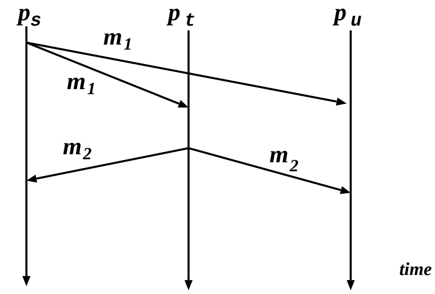

Figure 1: Causal precedence.

2.3 Distributed control

Every common destination of messages m_1 and m_2 is required to deliver m_1 before m_2 if m_1 causally precedes m_2. One way to realize the causally ordered delivery of messages is a *centralized* one where every process p_i sends a message to one controller and then the controller delivers the message to all the destination processes [Figure 2(1)]. Every process delivers messages in a same order as the controller receives the messages. Most distributed applications like teleconference systems take the centralized approach. This approach is simple and easy to implement. However, it takes at least two rounds to deliver a message to destination processes. The centralized approach is not suitable to realize real time, multimedia applications due to the long delay time. Another way is a *distributed* one where each process directly sends a message to destination processes. In addition, each process concurrently receives messages from multiple processes. It takes one round to deliver a message. In this paper, we take the distributed approach since it implies one round shorter delay time than the centralized approach. The *vector clock* [9] is widely used to causally order messages in distributed group protocols.

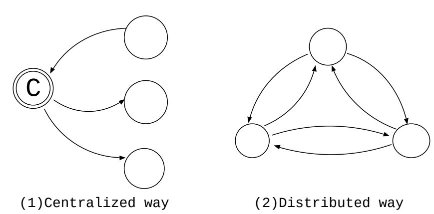

Figure 2: Data transmission.

2.4 O-precedency

We discuss how a pair of objects o_1 and o_2 transmitted by processes can be causally ordered. Let $ss_t(o)$ and $es_t(o)$ denote events that a process p_t starts and finishes sending an object o, respectively. In fact, $ss_t(o)$ and $es_t(o)$ show sending events that the top and last messages of the object o are sent by p_t, respectively. $sr_t(o)$ and $er_t(o)$ show receipt events of the top and last messages of the object o, respectively. Suppose a precess p_t receives an object o_1 and sends another object o_2. The object o_1 is *interleaved* with another

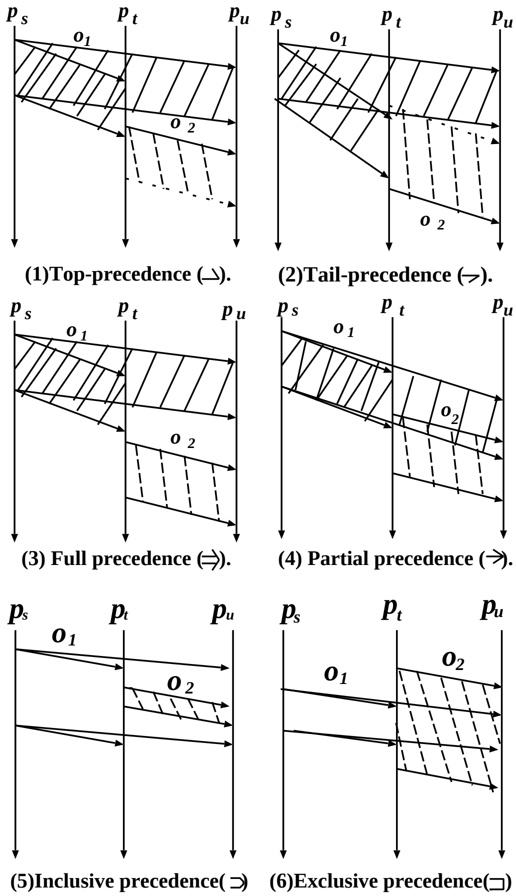

Figure 3: Precedency of objects.

object o_2 ($o_1 \| o_2$) iff $sr_t(o_1) \prec ss_t(o_2) \prec er_t(o_1)$ or $ss_t(o_2) \prec er_t(o_1) \prec es_t(o_2)$ in the process p_t. Here, the process p_t is receiving messages of the object o_1 while sending messages of o_2. Next, suppose p_t sends a pair of objects o_1 and o_2. o_1 is *interleaved* with o_2 in p_t ($o_1 \|_t o_2$) iff $ss_t(o_1) \prec ss_t(o_2) \prec es_t(o_1)$ or $ss_t(o_2) \prec ss_t(o_1) \prec es_t(o_2)$. The interleaving relation $\|_t$ is symmetric but not transitive. $o_1 \| o_2$ if $o_1 \|_t o_2$ in some process p_t.

[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 *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 ($o_1 \| o_2$).

4 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 $p_s = p_t$.

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

6 o_1 *exclusively precedes* o_2 ($o_1 \sqsupset o_2$) iff $o_1 \rightharpoondown o_2$ and $o_2 \rightharpoonup o_1$.□

An object o_1 $O-$*precedes* another object o_2 ($o_1 \rightsquigarrow o_2$) iff o_1 top, tail, fully, partially, inclusively, exclusively precede o_2. The logical properties on the O-precedent relation are discussed in a paper [14]. The COM protocol using two types of vector clocks is also presented in the paper [14].

3 QoS-based Precedency

3.1 Segments

The object(-O-)precedent relation does not imply how different it is between time when starting the transmission of o_1 and time when starting the transmission of o_2. A *synchronization* (*syn*) message is transmitted in order to synchronize communication of objects o_1 and o_2 at a smaller granularity level. A process sends a *syn* message each time the process sends some number of messages for each object. An object is partitioned into subsequences of messages which are named *segments*. Each segment starts at a *syn* message and ends at a next *syn* message. Suppose an object o is a sequence of messages $\langle$ ..., m_i, m_{i+1}, ..., m_j, ... $\rangle$ where m_i and m_j are *syn* messages and m_{i+1}, m_{i+2}, ..., m_{j-1} are normal messages. Here, a subsequence $\langle m_{i+1}, ..., m_j \rangle$ is a segment. If m_i is the top message m_1 of o_1, $\langle m_1, m_2, ..., m_j \rangle$ is a segment.

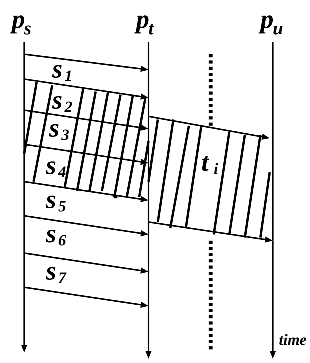

Figure 4: Precedency on segments.

An O-precedent relation on segments is defined in a same way as the O-precedent relations among objects. For example, a segment s_1 *fully precedes* another segment s_2 ($s_1 \Rightarrow s_2$) if a process starts sending s_2 after finishing receiving s_1.

[Definition] Let $\langle s_1, ... \rangle$ and $\langle t_1, ... \rangle$ be a pair of sequences of segments of objects o_s and o_t, respectively. Suppose o_s O-precedes o_t ($o_s \rightsquigarrow o_t$).

1. A subsequence $\langle s_i, s_{i+1}, ..., s_l \rangle$ of segments of o_s and a segment t_k of o_t are *synchronization blocks* iff s_{i-1} fully precedes t_k ($s_{i-1} \Rightarrow t_k$), s_i partially precedes t_k ($s_i \rightarrow t_k$), every s_h inclusively precedes t_k ($s_h \supset t_k$) (h=i+1, ..., l), and $s_{l+1} \not\supset t_k$.
2. A segment s_k of o_s and a subsequence $\langle t_i$, ..., $t_l \rangle$ of segments of o_t are *synchronization*

blocks iff s_k exclusively precedes t_h ($s_k \sqsupset t_h$) ($h = i$, ..., l-1), $s_k \not\sqsupset t_{i-1}$, and $s_k \rightarrow t_l$[Figure 5].□

Suppose a process sends six segments s_1, ..., s_6 of an object o to a pair of processes p_t and p_u and a process p_t sends a segment t_i of an object o_t to p_u as shown in Figure 4. Here, a sequence of segments $\langle s_2, s_3, s_4 \rangle$ should be synchronized with a segment t_i. Let o_s and o_t be objects where o_s O-precedes o_t ($o_s \rightsquigarrow o_t$). Let $\langle S_1$, ..., $S_n \rangle$ and $\langle T_1$, ..., $T_n \rangle$ be sequences of synchronization blocks of objects o_s and o_t, respectively. Here, each pair of blocks S_i and T_i are synchronization blocks. If each of synchronization blocks S_i and T_i includes one segment, a pair of objects o_s and o_t are referred to as *fully synchronized*.

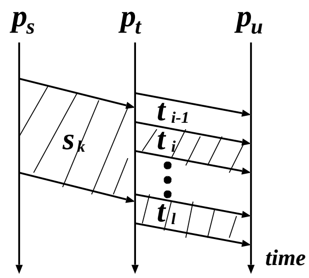

Figure 5: Synchronization blocks.

3.2 Synchronization of blocks

It is still question when a *syn* message to be transmitted. One way is that a sender process autonomously transmits *syn* messages independently of the other processes. Every common destination process p_u of objects o_s and o_t delivers messages so as to satisfy the O-precedent relation among synchronization blocks. This is referred to as *asynchronous* way for transmitting objects while receiving objects.

Another way is a *synchronus* way. For example, suppose a process p_s is receiving messages of an object o_s, another process p_t is sending messages of an object o_t, and o_s O-precedes o_t ($o_s \rightsquigarrow o_t$). The process p_t sends a *syn* message each time p_t receives a *syn* message from p_s. Here, the objects o_s and o_t are fully synchronized. In another example, the process p_t can send one *syn* message of the object o_t if p_t receives two segments from p_s. The process p_t can also send a pair of *syn* messages of o_t while p_t receives one segment. Thus, a pair of objects can be synchronized by sending *syn* messages.

Here, we introduce a *blocking factor* b_i for each pair of synchronization blocks S_i and T_i of objects o_s and o_t, respectively, where $o_s \rightsquigarrow o_t$. The blocking factor b_i is defined to be $\|T_i\|/\|S_i\|$. Here, a notation $\|B\|$ shows number of segments in a block B. If b_i=1, the objects o_s and o_t are fully synchronized. If $b_i > 1$, p_t sends more number of *syn* messages than p_s. If $b_i < 1$, p_t sends less number of *syn* messages than p_t.

Suppose that a process p_u receives a pair of segments s_1 and s_2 from processes p_s and p_t, respectively. The process p_t starts transmitting messages an object o_t after receiving a *syn* message from p_s. Then, p_t receives messages from p_s. In the meantime, if p_t receives *syn* message from p_s, p_t sends a *syn* message. Here o_s and o_t are fully synchronized. If s_1 fully precedes s_2 ($s_1 \Rightarrow s_2$), p_u is required to deliver all the messages in s_1 before s_2.

3.3 Quantity-based precedency

Each segment s of an object o is a subsequence of messages. A message is a unit of data transmission in the underlying networks. We assume each message has the same size. Here, let $g(s)$ show the number of messages included in the segment s. Let us consider a pair of objects o_1 and o_2 where o_1 O-precedes o_2 ($o_1 \rightsquigarrow o_2$) and o_2 is sent by a process p_t while o_1 is received by p_t, i.e. o_1 and o_2 are interleaved in p_t. The object o is decomposed into a sequence of segments s_{11}, s_{12}, ... and o_2 is also decomposed into a sequence of segments s_{21}, s_{22}, ... as shown in Figure 6. Suppose the objects o_1 and o_2 are fully synchronized and a pair of segments s_{1i} and s_{2i} are synchronization blocks (i = 1, 2, ...). Suppose a process p_s sends a video object o_1 to a pair of processes p_t and p_u and the process p_t sends a video object o_2 to the process p_u. The objects o_1 and o_2 are simultaneously displayed in p_u. Suppose that o_1 and o_2 have different frame rates f_1 and f_2, respectively. The synchronization factor $b_{12} = g(s_{2i})/g(s_{1i})$ is required to be f_2/f_1. Here, let b_{12} be a synchronization factor of the object o_2 to o_1. Thus, a process p_t is required to deliver a pair of objects o_1 and o_2 in the O-precedent relation $\rightsquigarrow$ if the objects o_1 and o_2 satisfy the QoS requirement at the process p_t.

[Definition] An object o_1 *quantity-precedes* another object o_2 with a blocking factor b ($o_1 \overset{b}{\rightsquigarrow} o_2$) iff $o_1 \rightsquigarrow o_2$ and the blocking factor of o_1 to o_2 is b.□

Suppose an object o_1 *quantity-precedes* another object with a blocking factor b ($o_1 \overset{b}{\rightsquigarrow} o_2$) where a process p_t sends o_2 while receiving o_1 from a process p_s. Suppose the process p_t receives a synchronization(*syn*) message m_1 from the process p_s and then receives messages in a segment s_1. The process p_t starts sending a new segment s_2 of the object o_2, i.e. p_t sends a *syn* message m_2. Then, p_t receives a *syn* message m_1 from p_s. Here, let $h(s_2)$ show the number of messages which p_t has sent so far. If $h(s_1)/g(s_2) \geq b$, p_t finishes sending s_2 by sending a *syn* message m_4. Suppose a process p_s sends an object o_1 to a pair of precesses p_t and p_u and the process p_t sends an object o_2 to p_u. Here, p_u receives o_1 and o_2. Suppose $o_1 \rightsquigarrow o_2$. If p_u delivers segments of o_1 and o_2 according to the causality of segments, o_1 and o_2 are delivered in p_u so as to satisfy $o_1 \overset{b}{\rightsquigarrow} o_2$. In Figure 6, a pair of the objects o_1 and o_2 are fully synchronized. Suppose a pair of segments s_{11} and s_{21} satisfy the QoS requirements $Q(o_1)$ and $Q(o_2)$. Messages of s_{11} and s_{21} are required to be delivered to p_u according to the O-precedent relation. Then, QoS of a segment s_{12} is degraded due to the channel congestion.

[Property] If $o_1 \overset{b_1}{\rightsquigarrow} o_2$ and $o_2 \overset{b_2}{\rightsquigarrow} o_3$, $o_1 \overset{b_3}{\rightsquigarrow} o_3$ where $b_3 = b_1 * b_2$. □

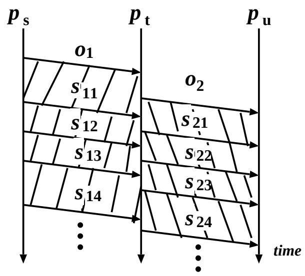

Figure 6: Synchronization of o_1 and o_2.

3.4 Quality-based precedency

Suppose an application is realized by a group of processes p_1, p_2, and p_3. Suppose a process p_1 sends an object o_1 to a pair of processes p_2 and p_3 and the process p_2 sends an object o_2 to p_3. The process p_3 receives the objects o_1 and o_2 in some type of O-precedent relation, i.e, $o_1 \rightsquigarrow o_2$. The application requires the processes to deliver each object o with QoS $Q(o)$ to destination processes in the group. Due to congestion in the networks, a process may not deliver messages of an object to a destination process with QoS required by the application. For example, suppose the bandwidth of a communication channel between p_1 and p_2 is degraded. Here the number of collars of the object o_1 is decreased to monochrome one by reduce the message size. If the application is not interested in the number of colors, the process p_1 can receive the object o_1 independently of the other object o_2. QoS of the segment s_{12} does not satisfy the QoS requirement $Q(o_1)$. Here, the process p_u can deliver messages of the segment s_{22} independently of messages of the segment s_{12}.

4 Application

We are now designing and implementing a teleconference system by using JGN(Japan Gigabit Network) [6]. At the 64th IPSJ annual conference held at Tokyo Denki University, Hatoyama, on March, 2002, a virtual session on high-level communication and applied technologies was held by cooperation of four sites, Iwate Prefectural Univ.(IPU), Tohoku Univ., Communications Research Lab.(CRL), and Tokyo Denki Univ.(TDU) which are interconnected in JGN [Figure 7]. Each site is composed of presentation device, video camera, and projector as shown in Figure 8. The video and voice data are transmitted by using IP, i.e. DV over IP [4]. In addition to transmitting multimedia data of video and voice of the conference, the powerpoint is used for presentation at each site. The data of the powerpoint is replicated in each site. The control signal to change pages is transmitted to all the sites without transmitting data in the pages. This virtual session was coordinated in a centralized controller of CRL. It takes two to four seconds to deliver video and voice due to the processing delay of each site. In order to overcome the difficulties, we are now taking the fully distributed approach discussed here to realizing the virtual conferences.

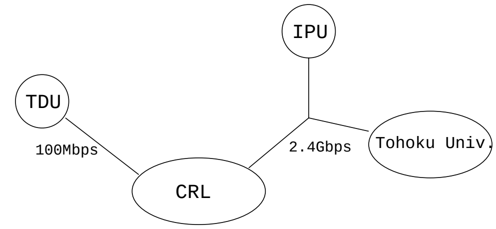

Figure 7: Network

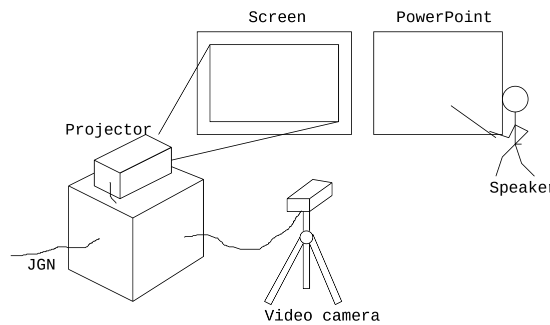

Figure 8:

5 Concluding Remarks

This paper discussed a group communications of multimedia objects with the fully distributed control. According to the O-precedent relations among multimedia objects, messages of the objects are delivered to destination processes ordered We discussed QoS-based precedency.

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] Shushi Uetsuki, Takahiro Komine, Akihiko Machizawa, Kazunori Sugiura, Michiaki Katsumoto, Shin-ichi Nakagawa, and Fumito Kubota, "Quality evaluation of DV over IP transmission reflection," Tech. Report of SIG Communication Quality, IEICE, 2000.

[5] Fischer, M. J., Lynch, N. A., Paterson, M. S., "Impossibility of Distributed Consensus with One Faulty Process," *Journal of the ACM (JACM)*, Vol.32, No.2, 1985, pp.374–382.

[6] JGN: Japan Gigabit Network. http://www.jgn.tao.go.jp/org_tec/index.html

[7] 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.

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

[9] 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.

[10] 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.

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

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

[13] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications," *RFC* 1889, 1996.

[14] Shimamura, K., Tanaka, K., and Takizawa, M., "Causally Ordered Delivery of Multimedia Objects," *Computer Communications Journal*, 2002, Vol. 25, No. 5, pp.437–444.

[15] Shimamura, K., Tanaka, K., Takizawa, M. : "Protocol for Synchronizing Multimedia Objects Exchanged in a Group of Process," *Journal of IPSJ*, 2002, Vol. 43, No. 2.

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

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

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