QoS Based Group Communication

5L−04

Seiichi Hatori, Kenichi Shimamura, and Makoto Takizawa
Tokyo Denki University
E-mail {hatori,ken,taki}@takilab.k.dendai.ac.jp

Abstract

In distributed applications, a group of multiple processes are cooperating by exchanging multimedia messages. The multimedia objects are longer than traditional messages. If messages transmitted in a network are causally delivered by using traditional group protocols, computation and communication overheads are increased due to large size of multimedia data. In this paper, we discuss a protocol to causally deliver multimedia objects.

1 Introduction

In distributed applications, a group of multiple processes are cooperating. In the group communication, a *group* is first established among multiple processes and then messages sent by processes are *causally* delivered to destination processes in the group. 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.

In distributed applications, various kinds of multimedia objects like image and video are exchanged among multiple processes in the group. Multimedia objects are structured and are larger than traditional data messages.

An object is decomposed into a sequence of messages in order to transmit the object in a network. A message is a unit of data transmission. If a pair of objects o_1 and o_2 are transmitted by processes p_1 and p_2. The messages decomposed from o_1 and o_2 are causally delivered to every common destination process p_3 of o_1 and o_2 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. In another application, the top message in a message sequence decomposed from an object o_1 is required to be delivered before the top message of another object o_2 while the other messages can be delivered in any order. Thus, we define new 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 destination. Only precedent relations among top messages of objects are discussed in the paper. In order to support QoS required by applications, message sequence of objects should be related at a smaller granularity. In the paper, we discuss how related granules of objects are in the networks.

2 System Model

Distributed applications are realized by cooperation of a group of multiple application processes. Application processes exchange objects including multimedia data with the other processes in the group by using the network. An application process is supported by a system process. The process p_t takes an object from the application process 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 among processes is a *message*. Messages are not lost and maximum delay time between every pair of processes is bounded in the network.

An object o is decomposed into a sequence of messages by a source process and the messages are delivered to the destination processes. Here, m_1 and m_k are the top and last messages of the object o. A destination process assembles received messages into an object and then delivers the object to the application process. The cooperation of the processes is coordinated by a *group protocol* which supports the reliable, efficient communication service by taking usage of the network service.

3 Object-Precedency

We discuss how a process sends and receives multimedia objects in a group G of multiple processes p_1, ..., p_n ($n > 1$). In order to increase the throughput and reduce the response time, sending and receiving events of objects are interleaved if there is no precedent relation among objects.

As presented here, a way on how a pair of objects o_1 and o_2 are interrelated depends on when processes p_s and p_t start and finish sending and receiving the objects. 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 a process p_t starts and finishes sending an object o, respectively. In fact, $ss_t(o)$ and $es_t(o)$ show events that the top and last messages 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.

[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 $o_2 \rightharpoonup o_1$.□

The top, tail, fully, partially, inclusively, and exclu-

sively precedent relations defined here are referred to as *object-causally precedent* (*O-precedent*) relation. Here, $o_1 \rightsquigarrow o_2$ shows that o_1 O-precedes o_2, i.e. $\rightsquigarrow \in \{\rightarrow, \rightarrow, \Rightarrow, \rightarrow, \supset, \sqsupset\}$. 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.

4 Granularity of Synchronization

4.1 Segments

A *synchronization* (*syn*) message is transmitted to synchronize communication of 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. Non-synchronization messages are referred to as *normal* ones. A sequence of messages of the object is decomposed into subsequences which are named *segments*. Each segment of messages starts at a *syn* message and ends at a next synchronization one. 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_{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 of o_1, $\langle m_i$, m_{i+1}, ..., $m_j\rangle$ is a segment. Thus, the object o can be considered to be a sequence of segments. A process p_s sends an object o_s to another process p_t. The object o_s is decomposed into a sequence of seven messages m_1, ..., m_7. Here, m_1, m_4, and m_7 are *syn* messages. In fact, m_1 and m_7 are the top and last messages of the object o_s, respectively. Thus, the object o_s is decomposed into a sequence of two segments s_1 and s_2 where $s_1 = \langle m_1$, m_2, m_3, $m_4\rangle$ and $s_2 = \langle m_5$, m_6, $m_7\rangle$.

[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), $s_k \not\supset t_{i-1}$, and $s_k \rightarrow t_l$.□

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*.

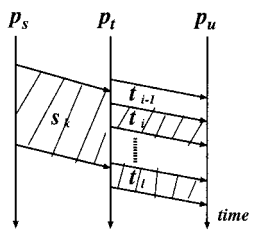

Figure 1: Synchronization blocks

4.2 Synchronization of blocks

A process p_t sends a *syn* message if some condition on messages which p_t is receiving is satisfied. For example, suppose a process p_s is receiving messages of an object o_s, 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, o_s and o_t are fully synchronized. The process p_t can send one *syn* message of o_t every two segments p_t receives from p_s. The process p_t can also send a pair of *syn messages* of o_t while p_t receives one segment. 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_j=1, o_s and o_t are fully synchronized. If $b_j > 1$, p_t sends more number of *syn* messages than p_s. Otherwise, p_t sends less number of *syn* messages than p_t.

4.3 QoS precedence

We assume each message has the same size. Here, let $g(s)$ show the number of messages included in the segment s. Suppose the objects o_1 and o_2 are fully synchronized and a pair of segments s_{1i} and s_{2i} are synchronization blocks. 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 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 blocking factor $b_{12} = g(s_{2i})/g(s_{1i})$ is required to be f_2/f_1. Here, let b_{12} be a blocking function of o_2 to o_2.

[Definition] An object o_1 *Q-precedes* another object o_2 ($o_1 \overset{Q}{\rightsquigarrow} o_2$) iff $o_1 \rightsquigarrow o_2$ and Q shows a blocking factor.□

Suppose an object o_1 *Q-precedes* another object o_2 ($o_1 \overset{Q}{\rightsquigarrow} o_2$) where a process p_t sends o_2 while receiving o_1 from a process p_s. Suppose p_t receives a synchronization(*syn*) messages m_1 from p_s and then receives messages in a segment s_1. The process p_t starts sending a new segment s_2 of 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 $h(s_1)/g(s_2) \geq Q$, p_t finishes sending s_2 by sending a *syn* message m_4. Suppose a process p_s sends 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 $o_1 \overset{Q}{\rightsquigarrow} o_2$.

[Definition] If $o_1 \overset{Q_1}{\rightsquigarrow} o_2$ and $o_2 \overset{Q_2}{\rightsquigarrow} o_3$, $o_1 \overset{Q}{\rightsquigarrow} o_3$ where $Q = Q_1 * Q_2$. □

5 Concluding Remarks

We defined the O-precedent relations among multimedia objects. In addition, we discussed synchronization (*syn*) messages in order to synchronize multiple objects which are O-preceded at a smaller granularity level.

References

[1] 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.