QoS Control in Group Communication

Takuya Tojo and Makoto Takizawa

Tokyo Denki University
E-mail {tojo, taki}@takilab.k.dendai.ac.jp

This paper discusses how to exchange multimedia messages in a group of multiple processes. Quality of Service (QoS) required by the applications has to be supported for multimedia applications. In traditional communication protocols like TCP and RTP, a process can reliably deliver messages to one or more than one process, i.e. one-way transmission. In the group communication, a process sends multimedia messages to multiple processes while receiving multimedia messages from multiple processes in a group. We discuss how to transmit multimedia messages to each destination process so as to satisfy QoS requirement among the processes.

グループ通信における QoS 制御

東條 琢也　滝沢 誠

東京電機大学理工学部情報システム工学科
E-mail {tojo, taki}@takilab.k.dendai.ac.jp

本論文では、複数のプロセスから構成されるグループ内のプロセス間で、マルチメディア・メッセージを通信する方法について論じる。アプリケーションによって必要とされるサービス品質 (QoS) をグループ内の各プロセスに提供せねばならない。グループ通信では、プロセスは複数のプロセスからメッセージを受信し、かつ複数のプロセスにメッセージを送信する。本論文では、グループ内のプロセス間に必要な QoS を満足するようにマルチメディア・メッセージを送信する方法について論じる。

1 Introduction

In distributed applications like teleconferences, a group of multiple processes are cooperating by exchanging messages. In group communication, a *group* of multiple processes is first established. Then, messages transmitted by processes have to be *causally delivered* to multiple destination processes in the group [3]. For example, a process p_1 sends a question message Q to a pair of processes p_2 and p_3. After receiving the question Q, the process p_2 sends an answer message A of the question Q to the processes p_1 and p_3. The message Q *causally precedes* the message A [3, 7, 8]. Here, the process p_3 is required to deliver the message Q before A. Thus, a message m_1 is referred to as *causally precede* another message m_2 if and only if (iff) a sending event of m_1 happens before a sending event of m_2 [3, 7, 8]. Various types of the group communication protocols which support a group of multiple processes with the causally ordered delivery of messages have been so far discussed [3,11].

In distributed applications, multimedia data is exchanged among processes in addition to traditional data in high-speed communication networks [6]. High-speed transmission protocols like XTP [5], and multimedia communication protocols like RTP [12] and RSVP [4] are developed so far, by which a large volume of multimedia data can be efficiently transmitted to one or more than one process. In these protocols, inter-message gap is controlled so that the buffer of the receiver does not overrun. Protocols to support Quality of Service (QoS) like delay time and message loss ratio are discussed [2, 6]. They discuss only one-to-one and one-to-many types of high-speed communications. Let us consider a teleconference which is composed of multiple remote sites. Video and voice of each remote site are distributed to every remote site in the teleconference. In one way, there is one centralized controller site. Every site first sends multimedia data to the controller. Then, the controller forwards the data to every remote site. The same video and voice data of every site is seen at every site. This is a centralized approach. This approach is simple and easy to implement the teleconference. However, it takes two rounds to deliver a message from a site to another site since every message is delivered through the centralized controller. The centralized way is not suited to realize real-time applications including multiple processes distributed in a wide-area network. We take a *distributed* approach where every process directly sends a message to destination processes in a group of processes in order to realize real-time constraints of multimedia data. Each process receives messages from multiple sites. Each process has to causally order messages received from multiple processes by itself in order to causally deliver the messages. In addition, a process is required to send a message to each destination process so that QoS requirement is satisfied. In this paper, we discuss a distributed group protocol for transmitting multimedia messages.

In section 2, we present a system model. In section 3, we discuss a model for transmission and receipt of multimedia messages in group communication. In section 4, we discuss a group protocol.

2 System Model

2.1 Channel

A group G of multiple processes $p_1, \ldots, p_n (n>1)$ are interconnected with reliable high-speed communication networks. The network is modeled to be a collection of reliable high-speed channels. Processes communicate with each other by taking usage of channels. There is a high-speed 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 $\langle p_i, p_j \rangle$ satisfies the following characteristics:

1. The channel is bidirectional, i.e. $\langle p_i, p_j \rangle$ exists if $\langle p_j, p_i \rangle$ exists.
2. The channel is reliable, i.e. any message is neither lost nor duplicated and messages are transmitted in a sending order in every channel.
3. The channel is high-speed, i.e. transmission time of a data unit is much shorter than the delay time.
4. Each channel is synchronous [10], i.e. the maximum delay time is bounded.

A process p_i sends a message m to one or more than one destination process in a group G. Let $dst(m)$ denote a collection of destination processes of a message m, which is a subset of a group G. Let $src(m)$ show a source process which sends a message m. A message m is transmitted from a process p_i to every destination process p_j in $dst(m)$ via a channel $\langle p_i, p_j \rangle$. Each channel $\langle p_i, p_j \rangle$ can be realized to be a connection like one supported by TCP [9].

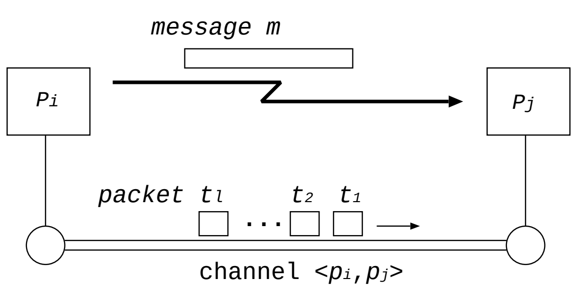

Figure 1: Channel

A process p_i sends a message m to destination processes in $dst(m) = \{p_{i1}, \ldots, p_{ik_i}\}$ $(k_i \geq 1)$. The message m is delivered to each destination process p_{ij} through a channel C_{ij}. A message is decomposed into smaller units named *packets* which are units of transmission in a channel. A sequence of packets are transmitted in a channel. Suppose a message m is decomposed into a sequence of packets $t_1, \ldots, t_l (l \geq 1)$[Figure 1]. Let $pkt(m)$ be a sequence of packets t_1, …, t_l of a message m. The process p_i transmits a packet sequence to every destination process p_{ik} via a channel $C_{ik} = \langle p_i, p_{ik} \rangle$. A destination process p_{ik} receives *packets* sent by the process p_i through the channel C_{ik} and assembles the packets into a message. Then, the message is delivered to the process.

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}. There are following QoS parameters for group communications among multiple processes:

1. bw:bandwidth [bps].
2. pk:packet loss ratio [%].
3. dl:delay [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, …, $m)$. Let Q be a set of QoS parameters q_1, …, 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 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 preference relation "$\rightarrow$" is a partially ordered relation on QoS parameters $q_1, \ldots, q_m$, i.e. $\rightarrow \subseteq Q^2$. "$q_i \rightarrow q_j$" shows that a parameter q_i is preferred to q_j by an application. The QoS parameters in Q are partially ordered in the preference relation "$\rightarrow$". For example, if *bw* is more significant than *pl* for an application, $bw \rightarrow pl$. 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 be a partially ordered set $\langle Q, \rightarrow \rangle$, named *preference* of an application. For example, $Q = \{bw, pl, dl\}$. An application specifies its precedent relation; $bw \rightarrow pl$ and $dl \rightarrow pl$ on Q. A preference P for an application is $\langle\ Q, \{bw \rightarrow pl, dl \rightarrow pl\}\ \rangle$.

Let A and B be QoS instances $\langle$128[Mbps], 100[msec], 0.05[%]$\rangle$ and $\langle$64[Mbps], 50[msec], 0.1[%]$\rangle$, respectively. Here, 64 $\succ$ 128[Mbps], 100 $\succ$ 50[msec], and 0.05 $\succ$ 0.1[%]. Since the bandwidth (*bw*) and delay time (*dl*) are more significant than the packet loss ratio (*pl*), $bw \rightarrow pl$ and $dl \rightarrow pl$ in a preference P. The QoS instance A is *more preferable than* B with respect to the preference P $(A \succ_P B)$ while the delay time of B is better than A. Let $Q(A)$ and $Q(B)$ show sets of QoS parameters of QoS instances A and B, respectively. Here, let A and B be QoS instances where $Q(A) = Q(B)$. A relation "$A \succ_P B$" is *inductively* defined as follows:

[Definition] A QoS instance A is *preferable* to another QoS instance B with respect to a preference P $(A \succeq_P B)$ iff

1. if A=$\{a_i\}$, B=$\{b_i\}$, and Q=$\{q_i\}$, $a_i \succeq_P b_i$.
2. if $A' = A$ - $\{a_i\}$, $B' = B$ - $\{b_i\}$, and $Q' = Q$ - $\{q_i\} = Q(A') = Q(B')$,
    - $A' \succeq_P B'$ if $q_i \rightarrow q$ for some QoS parameter q in Q'.
    - $A' \succeq_P B'$ and $a_i \succeq_P b_i$ if $q_i \nrightarrow q$ for every QoS parameter q in Q'.

3 Data Communication Model

3.1 Transmission

A process p_i sends a message m to every destination process in $dst(m)$. The message m is decomposed into a sequence of packets $t_1, \ldots, t_l$ ($l \geq 1$). A packet is a unit of data transmission in a network. There are following ways to transmit a packet sequence $pkt(m)$ ($=\langle t_1, \ldots, t_l\rangle$) to the destination processes [Figure 2]:

1. The 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). This is referred to as *synchronous* transmission of message m to multiple processes in $dst(m)$.
2. The process p_i sends a sequence $pkt(m)$ of the packets through each channel independently of the other channel. This is referred to as *asynchronous* transmission of m to multiple processes.

The synchronous transmission means *multicast* of each packet. Here, let *snd* (t, C) show a procedure to send a packet t through a channel C. The synchronous transmission can be realized by a following procedure:

for $h = 1, \ldots, l$

{ *snd* (t_h, C_{ij}); ...; *snd* (t_h, C_{ik_i});}

In the asynchronous transmission, a sequence of packets are transmitted for each channel. Let *Snd* (T, C) show a procedure to send a sequence T of packets $t_1, \ldots, t_l$ through a channel C, i.e. **for** h = 1, ..., l {*snd* (t_h, C);}. Here, a notation $F_1 \parallel F_2$ means that a pair of procedures F_1 and F_2 are concurrently performed. For example, $F_1 \parallel F_2$ is realized by creating a thread for each of F_1 and F_2. The asynchronous transmission can be realized by performing a following procedure:

Snd (T, C_{i1}) $\parallel \ldots \parallel$ *Snd* (T, C_{ik_i});

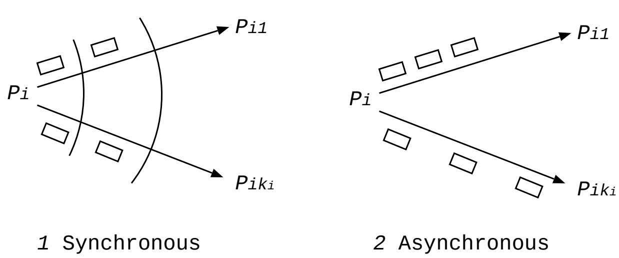

Figure 2: Transmission

Each destination process p_{ij} of a message m sent by a process p_i has some QoS requirement Q_{ij}. A process p_i has to deliver a message m to every destination process p_{ij} so as to satisfy the QoS requirement Q_{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 $p_1, \ldots, p_n$, every pair of processes p_i and p_{ij} do negotiation on the preference. Let P_{ij} denote a preference to be used when a process p_i sends messages to p_{ij}. Let a preference relation "$\succ_{ij}$" denote "$\succ_{P_{ij}}$". $Q_{ij}(t_k)$ is required to satisfy $Q_{ij}(Q_{ij}(t_k) \succ_{ij} Q_{ij})$. There are two cases:

1. For each packet t_h of a message m, $Q_{ij}(t_h) = \ldots = Q_{ik_i}(t_h)$. This is referred to as *quality-balanced* transmission of message m to multiple destination processes.
2. For some pair of channels C_{ij} and C_{ih}, $Q_{ij}(t_h) \neq Q_{ij}(t_k)$. This is referred to as *quality-unbalanced* transmission of m to multiple destination processes.

In the first case, each packet of a message m is sent with a same *QoS* in every channel. That is, a same packet is sent in every channel. In the second case, *QoS* in each channel is not necessarily same. A same packet is transmitted with different QoS instances in different channels.

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 *QoS-balanced* transmission. The QoS-balanced, synchronous transmission is referred to as *fully synchronous*. If some channel C_{ij} does not support enough *QoS*, e.g. due to congestion, 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 QoS-unbalanced, asynchronous transmission is referred to as *independent*. Figure 3 summarizes types of transmission.

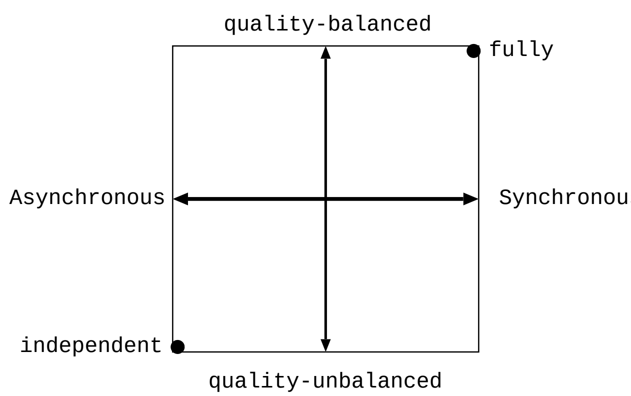

Figure 3: Types of transmission/receipt

3.2 Receipt

A process p_i receives messages from one or more than one process in a group G of processes $p_1, \ldots, p_n$. There are following ways for a process p_i to receive messages from multiple processes $p_{i1}, \ldots, p_{ik_t}$ ($k_i \geq 1$) [Figure 4]:

1. Each process p_{ij} sends a sequence $pkt(m_j)$ of packets $t_{j1}, \ldots, t_{jl_j}$ ($l_i \geq 1$) to a process p_i (j=1, $\ldots$, k_i). The process p_i receives a packet t_{jh} from each process p_{ij} after receiving a packet $t_{j,h-1}$ from every process p_{ij} (j=1, $\ldots$, k_i). This is referred to as *synchronous* receipt of messages from multiple processes.
2. A process p_i receives packets from each process p_{ij} independently of the other processes. That is referred to as *asynchronous* receipt of messages from multiple processes.

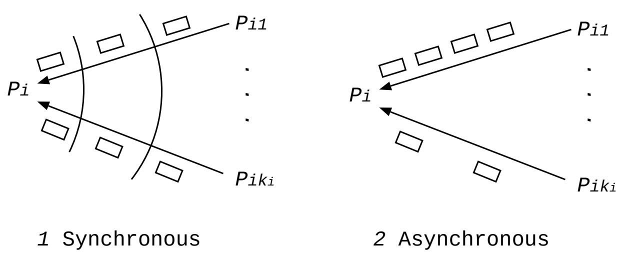

Figure 4: Types of receipt

Let "$t = rec\ (C)$" show a procedure to receive one packet through a channel C into a buffer t. Let T_j be a sequence of buffers $t_{j1}, \ldots, t_{jl}$, where each buffer can admit one packet. Let "$T_j = Rec(C)$" show a procedure to receive a sequence of packet into a buffer T_j; **for** h=1, $\ldots$, l $\{t_{1h} = rec(C)\}$. The synchronous and asynchronous receipts of a sequence of packets $t_1, \ldots, t_k$ are realized as follows:

1. Synchronous receipt:
   for $h = 1, \ldots, l$
   $\{ t_{1h} = rec\ (C_{i1}); \ldots t_{k_ih} = rec\ (C_{ik_i});\}$
2. Asynchronous receipt:
   $T_1 = Rec(C_{i1}) \parallel \ldots \parallel T_l = Rec(C_{el});$

As transmission of messages, there are following ways to receive messages from multiple processes:

1. A process p_i receives packets with a same QoS from each destination process p_{ij}. This is QoS-balanced receipt.
2. A process p_i receives packets with different QoS from different destinations. This is QoS-unbalanced receipt.

If a process synchronously receives messages in a QoS-balanced way, the process is referred to as *fully* receive messages. If a process asynchronously receives messages in a QoS-unbalanced way, the process is referred to as *independently* receive messages. A relation among types of receipt is shown in a same figure as Figure 3.

3.3 Communication

Each process sends messages to and receives messages from multiple processes in a group G. Much computation resource is spent to send and receive messages in each process. Since the computation resource is limited, each process may not send so many messages as the process would like to send and may not receive so many messages as the other processes send to the process. If more number of packets than a process can receive are arriving at the process, the process loses the packets. If the process spends much resource to receive messages, the process cannot synchronously send messages. Let $|t|$ show quantity of a packet t [bit]. If $Q_{ij}(t)$ is better than $Q_{ik}(t)$ ($Q_{ij}(t) \succ Q_{ik}(t)$), a packet t in a channel C_{ij} is larger than one in C_{ik}. Let $|t|_j$ show size of a packet t transmitted in a channel C_{ij}. In the QoS-balanced transmission, $|t|_j = |t|_k$ for a packet in every pair of channels C_{ij} and C_{ik}. Let $MaxQ_i$ denote maximum quantity of packets which a process p_i can send and receive in a second [bps]. For example, a pair of packets t_1 and t_2 are simultaneously sent to a process p_i from diferrent processes and the process p_i at the same time sends a packet t_3. If $|t_1| + |t_2| + |t_3| \leq MaxQ_i$, p_i can send and receive all the packets. Otherwise, p_i loses packets, e.g. due to buffer overrun or cannot send t_3 at a constant rate.

4 Protocol

4.1 Negotiation

A protocol is composed of two modules:

1. Negotiation
2. Transmission

First, every process in a group do negotiation on the preference. Each process p_i sends its preference P_i to all the other processes. Then, each process p_i obtains a same preference P in the group G. After negotiation, the processes start transmission of messages.

An application is assumed to be realized by cooperation of multiple processes $p_1, \ldots, p_n$ in a group G. We make following assumptions on communication in a group:

1. Every process sends a message to all the processes in the group G.
2. Every process is free from fault.

There are types of transmission and receipt of messages in a group, i.e. *synchronous* or *asynchronous*, *QoS-balanced*, or *QoS-unbalanced* ones. There are following cases depending on which types of transmission and receipt ways each process takes as shown in Table 1. For example, every process fully transmits messages and fully receives messages in case 1. Every process fully sends messages and asynchronously receives messages in case 4. Every process asynchronously sends messages and fully receives messages in case 13. Every process asynchronously sends messages and asynchronously receives messages in case 16.

4.2 Data transmission

We take a slow start strategy as taken in many protocols [5, 9]. In addition, we newly take a notification approach, where each process p_i notifies other

Table 1: Types of communication.

	transmission		receipt	
	synch	QoS	synch	QoS
1	◯	◯	◯	◯
2	◯	◯	◯	×
3	◯	◯	×	◯
4	◯	◯	×	×
5	◯	×	◯	◯
6	◯	×	◯	×
7	◯	×	×	◯
8	◯	×	×	×
9	×	◯	◯	◯
10	×	◯	◯	×
11	×	◯	×	◯
12	×	◯	×	×
13	×	×	◯	◯
14	×	×	◯	×
15	×	×	×	◯
16	×	×	×	×

synch
◯ : synchronous   × : asynchronous
QoS
◯ : QoS-balanced   × : QoS-unbalanced

processes of its QoS.

Let $\langle t_1, \ldots, t_l \rangle$ be a sequence of packets sent by a process p_s. For each packet t_i, $R(t_i)$ shows QoS required to receive a packet t_i. $Q(t_i)$ stands for QoS of a packet t_i. Each packet t_i carries QoS information $t_i.R$, $t_i.Q$, and $t_i.NR$. Here, $t_i.Q$ and $t_i.R$ indicate $Q(t_i)$ and $R(t_i)$, respectively. $t_i.NR$ shows $R(t_{i+k})$, QoS of a packet t_{i+k}. t_{i+k} shows a packet to be sent k packets after t_i. On receipt of a packet t_i from a process p_s, p_s get QoS information on t_{i+k} to be sent by p_s, $R(t_{i+k}) = t_i.NR$.

Before receiving a packet t_{i+k}, a process p_s starts receiving t_{i+k} on receipt of t_i. The process p_s may negotiate with other processes. That p_s can receive t_{i+k} in order to receive t_{i+k} so that QoS requirement t_i.NR is satisfied.

5 Concluding Remarks

This paper discusses how to exchange messages among multiple processes in a group so as to satisfy *QoS* required. We are now designing the protocol for exchanging multimedia data in a group of processes.

References

[1] Ahamad, M., Raynal, M., and Thia-Kime, G., "An Adaptive Protocol for Implementing Causally Consistent Distributed Services," *in Proc. of IEEE ICDCS-18*, 1998, pp.86–93.

[2] ATM Forum, "Traffic Management Specification Version4.0," 1996.

[3] Birman, K., "Lightweight Causal and Atomic Group Multicast," *ACM Trans. on Computer Systems,* Vol.9, No.3, 1991, pp.272–290.

[4] Braden, R., ed., "Resource ReSerVation Protocol," *RFC2205*, 1997.

[5] Chesson, G., "XTP/PE Overview," *Proc. of the IEEE 13th Conf. on Local Computer Networks*, 1988, pp.292–296.

[6] ITU-T I.361, "B-ISDN ATM Layer Specification," 1990.

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

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

[9] Marina del, R., "Transmission Control Protocol," *RFC793*, 1981.

[10] Michael, F., Nancy, L., and Michael, P., "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, Vol.32, 1985.

[11] Moser, L., Melliar-Smith, P. M., Koch, R., and Berket, K., "A Group Communication Protocol for CORBA," *Proc. of IEEE ICPP'99 Workshops*, 1999, pp.30–36.

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

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