2003－DPS－113 (4)
2003／6／19

Fair Communication in Multimedia Group Communication

Satoshi Itaya, Tomoya Enokido, and Makoto Takizawa
Tokyo Denki University, Japan
E-mail {itaya, eno, taki}@takilab.k.dendai.ac.jp

Abstract

This paper discusses a communication protocol for exchanging multimedia messages in a group of multiple peer processes. In group communication, a *group* of multiple processes are first established. Every process sends multimedia messages to multiple processes while receiving multimedia messages from multiple processes in a group. In addition, messages are required to be causally delivered. Since the computation resource like CPU are limited, each process may not receive all messages from other processes if the process spends much computation resource to receive all the messages from some process. Thus, it is significant to support each process with "fair" communication service in a group. In this paper, we discuss what the fairness means in a group of multiple processes interconnected with heterogeneous networks. We discuss a group protocol which supports a group of peer processes with fair communication.

マルチメディアグループ通信における公平通信モデル

板谷 智史　榎戸 智也　滝沢 誠
東京電機大学理工学部情報システム工学科
E-mail {itaya, eno, taki}@takilab.k.dendai.ac.jp

現在の情報システムはサーバ、ワークステーション、パーソナルコンピュータ、PDA 等の多種多様な端末から構成され、それぞれネットワークによって相互接続された分散型のシステムとなっている。電子会議や電子授業などの分散型アプリケーションは、複数のプロセスがグループを構成し、互いに映像、音声等のマルチメディアデータの送受信を行い、協調動作をなうことで実現されている。ネットワークの帯域幅や計算機の資源には限界があり、あるプロセスからのメッセージを受信するために資源を全て費す場合、他のプロセスから送信されたメッセージを受信できない場合がある。本論文では、ネットワークが提供するサービス品質および各プロセスの処理能力が異なる環境において「公平な通信サービス」を提供するグループ通信プロトコルを提案する。

1. Introduction

In distributed applications, multiple peer processes are cooperating by exchanging multimedia messages in high-speed communication networks. A process sends messages at such a rate that a destination process can receive the messages. There are many discussions on how to control the transmission rate so as to prevent from buffer overruns for one-to-one and one-to-many communications [1, 2, 14]. For example, a media server multicasts video data to multiple clients.

In group communication, a group is a collection of multiple peer processes and the processes exchange messages with each other, i.e. each process sends a massage to multiple destination processes while receiving messages from multiple processes [1,7]. In addition to reliably delivering messages to multiple processes, messages for multiple processes are required to be causally delivered [6, 7]. Each process can support the limited processing bandwidth. Hence, a process cannot receive all the messages transmitted in the network and cannot send as many messages as the process would like to send. For example, if a process spends most computation resource like CPU to receive all the messages from some process, the process cannot receive messages from other processes due to the lack of computation resource. Thus, processes are required to support applications with $fair$ group communication service in a group. In this paper, we define what the fairness means in group communication. We also discuss a group protocol to support a group of multiple peer application processes with fair communication.

In section 2, we discuss group communication service. In section 3, we discuss whether a traditional communication protocols can be used to realize group communication. In section 4, we discuss fair group communication protocol.

2. Group Communication

In group communications [9–11, 15], multiple processes p_1, ..., p_n (n>1) first establish a *group* and then the processes exchange messages with each other in the group. A process sends each message to multiple processes while receiving messages from multiple processes in a group. A group is a collection of peer processes where there is no centralized controller. Messages sent by a process are required to be reliably delivered to each destination process, i.e. without message loss and in a sending order. In addition, messages are required to be *causally delivered* in a group.

A message m_1 *causally precedes* another message m_2 ($m_1 \rightarrow m_2$) if and only if (iff) a sending event of m_1 *happens before* a sending event of m_2 [6, 11]. For example, suppose a process p_1 sends a message m_1 to a pair of processes p_2 and p_3 in a group. The process p_2 sends a message m_2 after receiving the message m_1. Here, m_1 causally precedes m_2 ($m_1 \rightarrow m_2$). Every common destination process p_1 of messages m_1 and m_2 is required to deliver m_1 before m_2 if m_1 causally precedes m_2.

There are a pair of approaches to realizing the group communication, *centralized* and *distributed* ones. In the centralized approach, there is one controller process which coordinates the cooperation of multiple processes in a group. Each process sends a message to the controller and then the controller forwards the message to all the destination processes. Current teleconference systems take the centralized approach. In the distributed approach, there is no controller and each process directly exchanges messages with the other processes. In this paper, we take the fully distributed approach to realizing the group communication.

A *group G* is composed of multiple peer processes p_1, ..., p_n (n>1). The processes communicate with each other by using logical channels supported by the underlying network. Each channel supports a pair of processes with some Quality of Service (QoS), i.e. bandwidth, packet loss ratio, and delay time. A message is delivered to each destination process through a channel. A process takes a multimedia message from an application process. The multimedia message is decomposed into a sequence of *packets* [Figure 1]. Packets are units of data transmission on an underlying network while a message is a unit of data transmission at application layer. The packets are transmitted to destination processes in a network. QoS supported by each channel is changing depending on types of underlying networks and due to congestions and faults of the underlying network.

A process is characterized by the total processing capacity [bps] to send and receive messages. If a process spends most computation resource to do some work, the process cannot do other works. For example, if a process spends the processing capacity to receive messages with high bandwidth from one process, the process may not receive messages from the other processes due to the lack of the processing capacity. Thus, each process cannot send and receive as many messages as the process desires. There, the process has to give up some desire. A pair of processes are to as *fair* if the rates of the desires which are satisfied are same in both processes.It is significant to support every process with fair communication in the group. There are many discussions on the fairness on service which process can take.

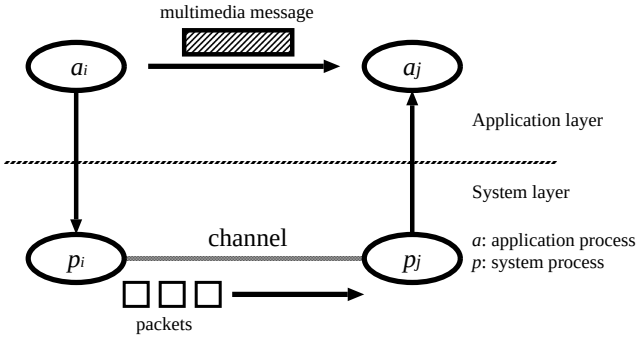

Figure 1. Communication model.

3 Traditional Communication Protocols

One approach to realizing group communication is to take usage of traditional protocol like TCP [12] and UDP [13]. In traditional connection-oriented protocols like TCP [12] and XTP [2], one-to-one and one-to-many communication are supported. The data transmission among a pair of processes is coordinated based on the feedback control. If a process fails to receive packets due to buffer overflow and overrun, the destination process informs the source process. On receipt of the notification, the source process decreases the transmission rate and retransmits the packets lost to the destination process. One the other hand, packets are just transmitted without any control in UDP [13]. However, some packets may be lost in the network. One way to realize a group protocol for exchanging multimedia messages is to take usage of UDP. We measure the receipt rate of packets in a pair of following configurations there process p_1, p_2, and p_3 as shown in Figure 2:

1. One sender process p_2 sends packets to a pair of receiver processes p_1 and p_3.
2. Two sender processes p_2 and p_3 send packets to one receiver process p_1.

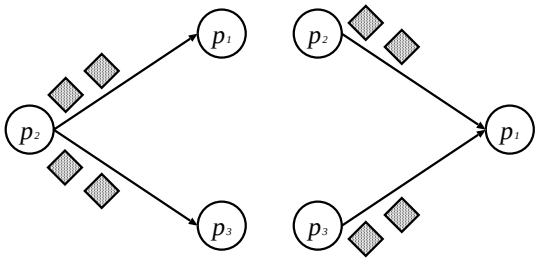

Figure 2. Evaluation system model.

The processes are realized in C language on the following computers which are interconnected with Gigabit Ethernet(1000BASE-SX):

p_1: Sun Enterprise 450 Server(Ultra SPARC-II 400MHz $\times$2, 1GB memory, Solaris 9).

p_2: Sun Blade 1000(Ultra SPARC-III 900MHz $\times$2, 2GB memory, Solaris 8).

p_3: Sun Enterprise 450 Server(Ultra SPARC-II 300MHz $\times$2, 592MB memory, Solaris 2.6).

In the first case, the source process p_2 continuously sends packets to a pair of destination processes p_1 and p_3 for 10 seconds at the transmission rate of 100[Mbps]. A pair of the destination processes p_1 and p_3 receive the packets sent by p_2. However, the receipt rates at the destination processes p_2 and p_3 are far time as shown in Figure 3. The receipt rate is smaller than the transmission rate, i.e. the destination process loses some packets.

In the second case, a pair of source processes p_2 and p_3 send packets with the transmission rate 80Mbps to the destination process p_1 for 10 seconds. Each of the source processes p_2 and p_3 transmits packets at rate 80[Mbps] for 10 seconds. However, the receipt rate of the destination process p_1 as slower than 80 [Mbps] as shown in Figure 4. Due to the limited processing rate, there are more number of packets which the process p_1 cannot receive.

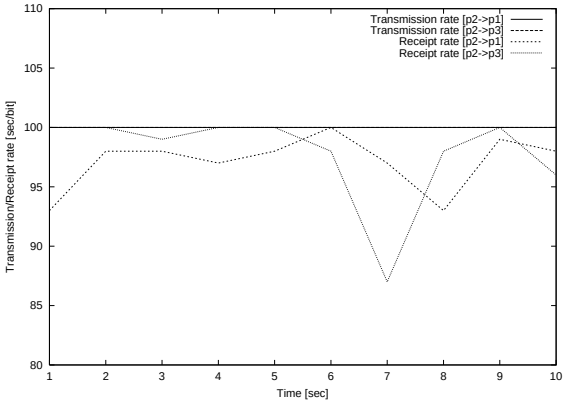

Figure 3. Evaluation (1).

The network bandwidth and the processing capacity of each process are limited. In order to realize fair communication, each process has to allocates computation resource like CPU to receive and send messages in the network.

4. Fair Communication

4.1. Model

We consider a group of multiple peer processes p_1, ..., p_n (n>1)which exchange multimedia messages. The network bandwidth and the processing capacity

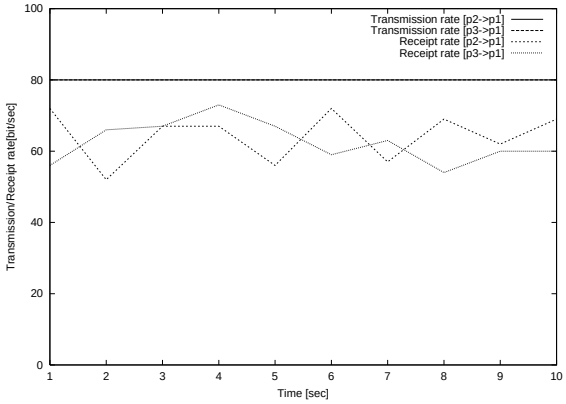

Figure 4. Evaluation (2).

[bps] of each process are limited. The total processing capacity should be fairly to send and receive messages to and from each process, respectively. In order to realize fair communication, each process has to allocates computation resource like CPU to receive and send messages in the network. The maximum processing rate of each process is bounded. If a process consumes most portion of total processing capacity to receive messages from another process, the process cannot receive messages from other processes. The system has to support a group of multiple processes with fair communication service. The fairness is measured in terms of QoS with which each application requires to be supported. There are discussions on how to define the fairness for multiple peer processes in a group. We consider a system which supports a group of processes with fair communication in following environment:

1. Every process in a group is sending messages at a same transmission rate.
2. It takes each process same time to transmit a unit of application data.
3. Every process in a group is sending messages at a same throughput.

We assume that the underlying network is reliable and synchronous, i.e. no packet is lost and delay time is bounded. Let δ_{ij} be delay time between a pair of processes p_i and p_j. RTT shows round trip time, which is given as $2\delta_{ij}$ for every pair of processes p_i and p_j in a group [Figure 5]. Some portion α_i of the total processing capacity is saved at each process p_i in a group. A process which newly transmits messages starts sending the messages to the process by using the portion α_i. We show parameters to discuss fairness to hold among processes in a group for exchanging messages.

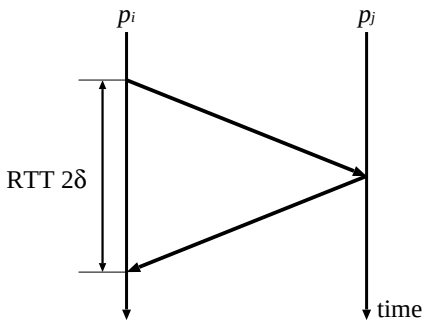

Figure 5. Round trip time (RTT).

$MRR_i[t]$ = maximum receipt rate of a process p_i at time t[bps].
$TR_{ij}[t]$ = transmission rate at which p_i sends packets to p_j at time t [bps].
$TR_i[t]$ = total transmission rate of p_j [bps] = $TR_{i1}[t]$+ ... + $TR_{in}[t]$.
$RR_{ij}[t]$ = receipt rate at which p_i receives packets from p_j at time t [bps].
$RR_i[t]$ = total receipt rate of p_i at time t [bps] = $RR_{i1}[t]$ + ... + $RR_{in}[t]$.

Since it takes δ_{ij} time units to deliver a packet from p_i to p_j, there are following relations among receipt rates and transmission rates:

$RR_{ij}[t] = TR_{ji}[t - \delta_{ij}]$.
$RR_i[t] = TR_{1i}[t - \delta_{1i}] + \ldots + TR_{ni}[t - \delta_{ni}]$.

If the total receipt rate $RR_i[t]$ gets larger than the maximum receipt rate $MRR_i[t]$ at time t, the buffer overruns in a process p_i. Here, $RR_i[t] \leq MRR_i[t]$ is required to hold in each process p_i.

A process p_i spends the processing capacity to receive and send packets:

$TC_i[t]$ = total processing rate at time t [bps].
$C_i[t]$ = processing rate at time t [bps] = $C_T \times TR_i[t] + C_R \times RR_i[t]$. Here, C_T and C_R are constants.

If $C_i[t] \leq TC_i[t]$, the process p_i can send and receive packets without delay and packet loss. Otherwise, the process p_i cannot send and receive all messages since, there is no enough processing capacity to do. Hence, the transmission rate of a source process has to decreased or the process p_i decreases the transmission rate. We take *slow start* and *traffic estimation* (SSTE) approach as the transmission rate control.

4.2 Transmission rate control

Multiple peer processes are sending multimedia messages to multiple processes while receiving multimedia messages from multiple processes in a group. Each process sends multimedia messages with transmission rate control. We assume $TC_i = TC_i[t]$ for every t. We also assume the $\alpha_i = \alpha$ for every process p_i and $C_T = C_R = 1$. Let p denote number of source process or destination processes. First, a source process sends messages by the slow start algorithm. The transmission rate is increased 10 [%] for every time unit. The faster the process p_i transmits messages, the larger capacity p_i spends. Hence, if the total processing rate $C_i[t]$ (= $TR_i[t]$ + $RR_i[t]$) gets equal to (TC_i - α), the process p_i does not increase the transmission rate $TR_i[t]$, i.e. $TR_i[t]$ + $RR_i[t]$ = (TC_i - α) [bps], $TR_i[t+1] = TR_i[t]$. Here, the process p_i has still some portion α of the processing capacity which is not used. If another process p_j would like to newly start to transmit messages, the process p_j sends messages to p_i by using the rate α.

[Transmission and Receipt messages]
for t=1, ..., k {
  $TR_{ij}[t+1] := TR_{ij}[t] \times 1.10$.
  if($TR_{ij}[t] + RR_{ij}[t]$) = (TC_i - α) then
    $TR_{ij}[t+1] := TR_{ij}[t]$.
  if(**receive new message**) then {
    if(**increase rate**) then
      $TR_{ji}[t+l] := TR_{ji}[t] + (\frac{\alpha}{2\delta}l)/p$ ($l \leq 2\delta$).
      if $TR_{ji}[t] = \frac{\alpha}{p}$ then $TR_{ji}[t+1] := TR_{ji}[t]$.
    if(**decrease rate**) then
      $TR_{ij}[t+l] := TR_{ij}[t] - (\frac{\alpha}{2\delta}l)/p$ ($l \leq 2\delta$).
  }
}□

We take the traffic estimation strategy to prevent buffer overruns. Suppose that the buffer overruns in a process p_i. The process p_i sends QoS requirement to every source process which is sending messages to p_i. Let $m.rq[bw]$ denote a *QoS requirement* message, QoS requirement indicates bandwidth required by the destination process to the source process. When a source process p_i receives QoS requirement, process p_j sends a message at bandwidth required by the process p_i. In a process, the buffer is assumed to overruns, if the traffic of a certain fixed quantity is exceeded. We show the traffic estimation procedure and sending QoS requirement as follows.

[Traffic estimation]
for t=1, ..., k {
  $traffic[t] := TR_i[t] + RR_i[t]$.
  if($traffic[t] > (TC_i - \frac{\alpha(p-1)}{p})$) then {
    $bw := TR_{ij}[t] \times (\frac{TR_{ij}[t]}{TR_i[t]+RR_i[t]})$.

send(**m.rq[bw]**).
}
}

for t=1, ..., k {
if(**receive m.rq[bw]**) then
$TR_{ij}[t+1] := bw$
}□

4.3 Fairness based on quantity of data to be transmitted

In multimedia communication, a multimedia message is longer than a traditional message and it takes time to transmit the message. Thus, fairness can be defined paying attention to quantity of data to be transmitted. Let Ts_{ij} denote a the size of data which a process p_i sends to a process p_j. A message is decomposed into packets Ps.

$$Ts_{ij} = \sum_{t=1}^{k} Ps$$

Ts_{ij} : quantity of data to be transmitted.
k : number of transmission packets.
Ps : transmission packet size.

Each process is fair if each process sends a same amount of data at a time. In multimedia group communication, a sender process sends a multimedia message m with same data size to multiple processes. However, some process may not receive all the messages due to the limited processing capacity. A process broadcasts information on quantity of data to send at the fixed interval in a group. Then, each process compares the quantity information with its information. Each process is fair if the same quantity of data is transmitted and received.

4.4 Fairness based on priority of sender process

Let us consider type of fairness based on priority of sender process. In multimedia group communication, a process sends multimedia messages to multiple processes while receiving multimedia messages from multiple processes. A process must deliver messages to a application process. Fairness is maintained when a receiver process delivers messages to application based on a priority of sender processes. We consider that the difference of bandwidth is able to mitigate by delivering messages to application based on a priority. Multimedia message is decomposed into a sequence of packets. A process delivers packets received from multiple processes based on priority of sender processes.

Each of packets has priority parameter. A receiver process schedules to delivering of packets with the priority parameter. Packets received are scheduled to be delivered as follows:

1. Delivers the higher of the priority parameter of a packet.
2. If the holding time in the queue gets longer than the fixed time, a priority will be increased and it deliver to application.

Then if holding time in queue as longer, packet will become unfair irrespective of priority of sender processes. The priority of the transmitted packet may be changed in a receiver process. If a priority is not changed by the holding time in the queue, the packet may not be delivered to application for a long time.

5. Concluding Remarks

This paper discussed fair communication model in multimedia group communication. We showed fair communication model with three types of fairness. Future work, we design detailed algorithm for fair communication.

References

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

[2] G. Chesson. XTP/PE Overview. *Proc. of the IEEE 13th conf. on Local Computer Network*, pages 292-296, 1988.

[3] IEEE. Standard for 1000BASE-T, 802.3ab. 1999.

[4] IEEE. Standard for Gigabit Ethernet, 802.3z. 1998.

[5] IEEE. Standard for Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications-Media Access Control(MAC) Parameters, Physical Layer and Management Parameters for 10 Gb/s Operation. 2002.

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

[7] F. Mattern. Virtual Time and Global States of Distributed Systems. *Parallel and Distributed Algorithms*, pages 215-226, 1989.

[8] A. Nakamura, and M. Takizawa. Run Synchronization in the Priority-Based Broadcast Protocol. *IPSJ DPS58*, pages 107-114, 1992.

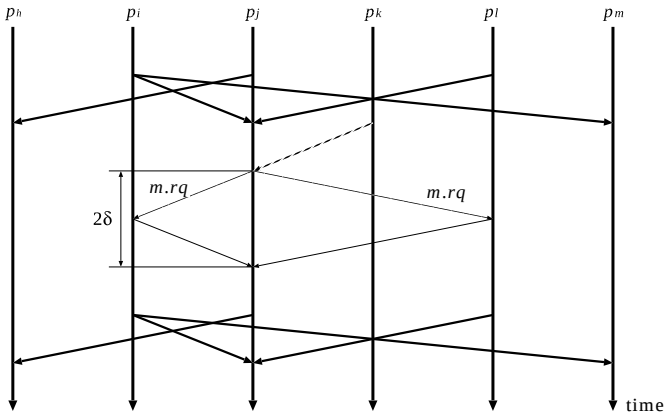

Figure 6. Exchanging messages.

[9] A. Nakamura, and M. Takizawa. Rate-Based Flow Control Schemes for High-Speed Group Communication. *Proc. of the Int'l Conf. on Parallel and Distributed Systems(ICPADS'93)*, pages 632-636, 1993.

[10] A. Nakamura. Reliable Broadcast Communication Protocols for Distributed Systems. Doctoral Dissertation, Tokyo Denki University, pages 71-78, 1994.

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

[12] J. Postel. Transmission Control Protocol. *RFC793*, 1981.

[13] J. Postel. User Datagram Protocol. *RFC768*, 1980.

[14] H. Schuizrinne, S. Casner, R. Frederick, and V. Jacobson. RTP:A Transport Protocol for Real Time Applications. *RFC1889*, pages 22-29, 1996.

[15] T. Tachikawa, and H. Higaki, and M. Takizawa. Group Communication Protocol for Realtime Applications *Proc. of IEEE ICDCS-18*, pages 40-47, 1998.