Flexible Group Communication Protocol

3L－07

Roziali Ghopur and Makoto Takizawa

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

Abstract

In a distributed application, a group of multiple processes is required to be cooperating by exchanging multimedia data. We discuss a group communication protocol which is so flexible that efficient communication is supported for each application in a given network service.

1 Introduction

In group communication, multiple processes establish a *group* and then messages are exchanged among processes. In the group, a process sends a message to multiple processes and receives messages from multiple processes. Messages are required to be causally delivered to processes in the group. Messages sent by a process may be lost due to conjestions and fault in the network. A group communication protocol is composed of following fuctions:

1. Transmission of message to multiple processes.
2. Confirmation of message receipt.
3. Detection of message loss.
4. Retransmission of a message.

There are various ways for transmission, comfirmation, detection, and retransmission. In addition, group protocol depends on what kinds of communication service the underlying network supports for processes. In this paper, we discuss a flexible group protocol where functions can be selected in given underlying network service so as to satisfy the requirement.

In section 2, we present a model of underlying network service. In section 3, we discuss service to be suppoted by group communication. In section 4, we discuss what types of functions to be selected to design a group communication protocol.

2 Underlying Network

A group of processes p_1, ..., p_n are cooperating through exchanging messages in the underlying network service. There are one-to-one network and broadcast network. In the one-to-one network, a message is sent to one process. A TCP connection supports one-to-one communication service. In the broadcast network, a message is sent to all the processes. The Ethernet and radio networks support the broadcast network. In the reliable one-to-one network, messages are delivered to the destinations with neither message loss nor duplication and in the sending order. In a reliable broadcast network, every process receives messages in the same order. In a less reliable network, messages maybe lost.

3 Group Communication Service

In group communication, a group of multiple processes p_1, ..., $p_n (n>1)$ are exchanging messages in the network. Let $s_i(m)$ and $r_i(m)$ denote sending and receipt events of a message m in a process p_i. A message m_1 *causally precedes* another message m_2 ($m_1 \rightarrow m_2$) if and only if (iff) $s_i(m)$ happens before $r_j(m)$. For example, suppose there are three processes p_1, p_2, and p_3 in a group G. A process p_1 sends a message m_1 to a pair of processes p_2 and p_3. The process p_2 sends a message m_2 to p_3 after receiving a message m_1. Here, m_1 causally precedes m_2 ($m_1 \rightarrow m_2$). Due to communication delay, m_1 may arrive at the process p_3 after m_2. p_3 is required to receive m_1 before m_2 because $m_1 \rightarrow m_2$. In order to causally deliver messages, each process p_i manipulates a vector $V = \langle V_1, ..., V_n \rangle$. Initially every element in the vector is zero. Each time a process p_i sends a message m, $V_i := V_1 + 1$. Then a message m carries the vector clock $V(m.V)$. On receipt of a message m, the vector clock $V = \langle V_1, ..., V_n \rangle$ is manipulated by a process p_i as follows: $V_j :=$ max$(V_j, m.V_j)$ ($j = 1,...,n$, $j \neq i$). A message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$) iff $m_1.V < m_2.V$. m_1 is *causally concurrent* with m_2 if neither $m_1 \rightarrow m_2$ nor $m_2 \rightarrow m_1$.

4 Group Protocol

4.1 Control

It is significant to discuss which process coordinates cooperation of multiple processes in a group. There are following types of control:

1. Centralized control.
2. Distributed control.

In the centralized control, there is one centralized controller in the group. A process first sends a message to a controller and then the controller reliably forwards the message to the destination processes. Each destination process sends receipt confirmation to the controller if the process successfully receives the message. Then the controller sends receipt confirmation to the sender if the controller receives the confirmation message from all the destination process. On the other hand, there is no centralized controller in the distributed control scheme. Each process makes a decision of correct receipt by itself.

4.2 Transmission

We discuss how to deliver a message to multiple destinations in a group. There are following ways to transmit a message m to multiple process:

1. Direct transmission.
2. Indirect transmission.

In the direct transmission, each process directly sends a message to each destination. In the indirect transmission, messages may be sent to some process and then the process forwards the message to the destination processes. Tree routing is an example of the indirect transmission. In the centralized control, the indirect transmission is adopted. In the direct transmission, it takes one round to deliver a message to a destination process. It takes more than one round to deliver a message in the indirect one. The direct one is preferable in real-time communication.

4.3 Confirmation

After a process sends a message, a destination process has to confirm other processes of the correct receipt of message. There are following ways to confirm the message receipt[Figure 1]:

1. Centralized confirmation.
2. Decentralized confirmation.
3. Distributed confirmation.

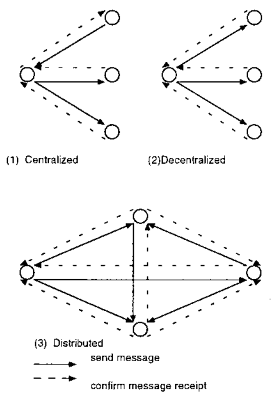

Figure 1: Message confirmation

In the distributed confirmation, each destination process sends a receipt confirmation to not only the sender process but also all the other destination processes. Since a group includes n processes, a sender process p_i sends (n - 1) messages in the one-to-one network and one message in the broadcast network. Then each destination sends (n - 1) confirmation messages in the one-to-one network and one confirmation message in the broadcast network. Hence totally $(n-1)^2$ and (n - 1) messages are transmitted in the one-to-one and broadcast networks, respectively. In the one-to-one network, communication overhead is (n^2) for number n of processes. In order to reduce number of messages transmitted in the network, confirmation information of message receipt is carried back and delayed confirmation strategy is adopted.

Suppose a process p_i sends a message to processes p_1, ..., p_n. In the centralized confirmation, every destination process p_j sends a confirmation message to one controller p_k if p_j is succeded in receiving the message m. If p_k receives confirmation messages from all the destination processes, the process p_k sends a confirmation message to p_i. In the decentralized confirmation, each destination process p_j sends a confirmation message back to a sender process p_i of the message m. Each process p_i does not send only confirmation message each time the process receives a message. If there is no data to be sent, p_i sends a confirmation of the messages to each destination process of the message, after p_i receives some number of messages.

4.4 Detection of message loss

Messages are lost due to buffer overruns, unexpected delay, and conjestion. Hence ,the processes have to recover from the message loss. We can detect message loss by checking sequence number of message and receipt confirmation. In the distributed confirmation, a process p_i can find loss of message from another process p_j by receiving a message from p_k.

4.5 Retransmission

A process p_i may fail to receive message. If p_i is detected to fail to receive message m, the message m will be retransmitted. There are following retransmission ways[Figure 2]:

1. Sender retransmission.
2. Destination retransmission.

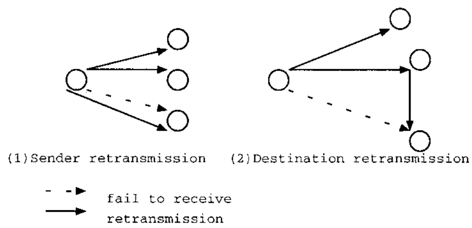

Figure 2: Retransmission

5 Concluding Remarks

In this paper, we made clear what types of functions to be realized in group communication protocol. We are now discussing how to select function so as to satisfy requirements.

References

[1] Lamport, R., "Time, Clocks, and the Ordering of Events in Distributed Systems," Communication of the ACM, Vol.21, No.7,pp 558-565, 1978.

[2] Takizawa, M.,"Design of Highly Reliable Broadcast Communication Protocol," Proc. of the IEEE COMPSAC87, pp.731-740, Tokyo, Oct. 1987.

[3] Takizawa, M. and Nakamura, A., "Reliable Broadcast Communication," Proc. of IPSJ Inter. Conf. on Information Technology (InfoJapan), pp 325-332, 1990.