Membership Management of Group

3L−06

Masashi Shiraishi and Makoto Takizawa
Tokyo Denki University, Japan
Email: {shira, taki}@takilab.k.dendai.ac.jp

Abstract

This paper discusses a group communication protocol where membership is dynamically changed due to faulty processes. We discuss how to support causally ordered delay of messages on less reliable networks.

1 Introduction

In a distributed application, a group of multiple processes are cooperating to achieve some objectives. Messages transmitted are causally / totally delivered to processes in the group [2]. A message m_1 *causally precedes* another message m_2 if a sending event of m_1 *happens before* [1] a sending event of m_2. If m_1 causally precedes m_2, m_1 is required to be delivered before m_2 in every common destination of m_1 and m_2.

Some process in a group may be faulty. Another process may newly join a group. Thus, a membership of group is variant, where processes are leaving and joining. In this paper, each communication channel between a pair of processes is assumed to be less-reliable. A process p_i perceives another process p_i to be faulty if p_i does not receive any message for some duration. When a process recovers from the fault, the process informs all the operational members of its recovery. Each process p_i has a *view* v_i showing what process is perceived to be operational in a group. The view v_i does not include a process p_j if p_i perceives p_j to be faulty. If p_i perceives some change of the membership, i.e. leaving and joining the group, v_i is changed. However, another process p_i might perceive p_j still operational. Here, $v_i \neq v_j$.

In most discussions on membership management [3], networks are assumed to be reliable. In addition, some mechanism for detecting faulty processes is assumed to be supported by the underlying subsystem. The underlying system is realized by taking usage of the Internet and mobile communication. In these systems, communication channels like connections supported by TCP are often disconnected while processes are still operational. Suppose there are three processes p_i, p_j, and p_k. A channel between p_i and p_j is eventually disconnected due to timeout while a channel between p_j and p_k is connected. Here, p_i considers p_j to be faulty because p_i cannot receive any message from p_j but p_k considers p_i to be operational. We discuss a group communication of multiple processes where processes are leaving and joining. In this paper, we discuss how to detect faulty processes.

In section 2, we discuss a system model. In section 3, we discuss how to detect faulty processes.

2 System Model

A group G is a collection of process p_1, ..., p_n ($n > 1$) which are interconnected in a network. The network is modeled to be a collection of channels. Each channel c_{ij} supports reliable, bidirectional communication between a pair of processes p_i and p_j. For example, a channel can be realized by a TCP connection. Messages sent by a process p_i are delivered to the other process p_j in a sending order without message loss as long as a channel c_{ij} exists. For example, if a connection is disconnected, p_i and p_j cannot be communicated. Even if p_i does not receive any message from p_j, p_i cannot perceive p_j to be faulty. Here, p_j is *suspected* by p_i.

A process is assumed to suffer from only crash fault. No Byzantine faulty occurs. In a network, each channel is assumed to be reliable, i.e. messages are delivered in a sending order without any message loss and duplication. However, a channel may be disconnected in the Internet and mobile networks. If a process is faulty, channels with the faulty process are automatically disconnected. In Figure 1, there are three processes p_1, p_2, and p_3. If the process p_3 is faulty, a pair of channels c_{13} and c_{23} are disconnected [Figure 1(2)]. On the other hand, a channel may be disconnected while processes interconnected with the channels are operational. For example, a channel c_{13} is disconnected while the processes p_1 and p_3 are operational [Figure 1(3)]. A process p_i cannot communicate with another process p_j if a channel between p_i and p_j is disconnected. Here, p_i perceives that p_j is suspected. p_i perceives p_j to be faulty if every operational process for p_i perceives p_j to be suspected or faulty.

Processes exchange messages in the network. Let $s_i(m)$ and $r_i(m)$ be sending and receipt events of a message m at a process p_i. The causally precedent relation is defined by the happens-before relation [1]. A message m_1 is referred to as *causally precede* another message m_2 ($m_1 \rightarrow m_2$) iff $s_i(m_1)$ *happens before* $s_j(m_2)$. m_1 is *causally concurrent* with m_2 ($m_1 \parallel m_2$) iff neither $m_1 \rightarrow m_2$ nor $m_2 \rightarrow m_1$. A message m_1 *totally precedes* another message m_2 iff $m_1 \rightarrow m_2$ or m_1 precedes m_2 in

every pair of common destination process of m_1 and m_2 if $m_1 \parallel m_2$.

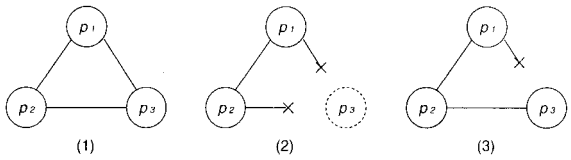

Figure 1: Disconnection of channel.

3 Detection of Faulty Process

As discussed in the preceding section, a process p_i cannot communicate with another process p_j if a channel c_{ij} between p_i and p_j is disconnected even if p_j is still operational. A process p_i considers another process p_j to be *operational* if p_i receives some message from p_j. A process p_i suspects p_j if p_i does not receive any message for a longer duration than some prefixed time units. An operational process p_i perceives another process p_j to be faulty if p_j is suspected by every process p_k which p_i is perceived to be operational. Let us consider processes in Figure 1(3). A pair of processes p_1 and p_2 cannot communicate with one another due to the disconnection of a channel c_{13}. However, p_1 can still deliver messages to p_3 via p_2. If p_i finds that p_i can deliver messages to a suspected process p_j via other processes, p_j is $semi-operational$. Figure 2 shows a state transaction showing how a process perceives another process.

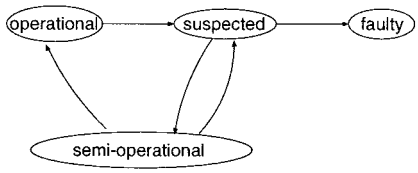

Figure 2: State transaction.

Each message m has a sequence number $m.sq$. The sequence number sq is incremented by one each time a process p_i sends a message. Each process p_i sends a message m with a vector $\langle sq_1, ..., sq_n \rangle$ to all the processes where each sq_j shows a sequence number of message which p_i expects to receive next from a process p_j. Suppose a process p_i receives a message m from another process p_j. Here, p_i finds that p_j has received every message m_i from p_i where $m_i.sq \leq m.sq_i$. In addition, the process p_i knows that p_j has received every message m_k from a process p_k when $m_k.sq \leq m.sq_k$. If $sq_j + 1 = m.sq$, p_i accepts m from p_j. If $sq_j > m.sq$, p_i rejects m. If $sq_j + 1 < m.sq$, p_i finds there is a message gap from p_j. Suppose $sq_k < m.sq_k$, p_i finds that p_i has not received message m_k from p_k where $sq_k < m_k.sq_k \leq m.sq_k$.

In Figure 3, a process p_1 sends a message m_1 with a vector $\langle 2, 1, 3 \rangle$ to a pair of processes p_2 and p_3. However, p_3 does not receive m_1 due to the disconnection of the channel c_{13} while p_2 receives m_1. Then, the process p_2 sends a message m_2 with $\langle 2, 2, 3 \rangle$ to p_1 and p_3. p_3 receives m_2 and then sends a message m_3 with $\langle 2, 2, 4 \rangle$ to p_1 and p_2. The process p_2 sends a message m_4 with $\langle 2, 3, 4 \rangle$ to p_1 and p_3. After sending m_1 to p_3, p_1 does not receive any message from p_3. p_1 suspects p_3. On receipt of m_4, p_1 finds that p_2 receives some message from p_3 but p_1 does not receive. That is, p_1 finds that p_2 perceives p_3 to be operational. Here, p_3 is semi-operational in p_1. p_2 forwards p_3 a message which p_1 sends.

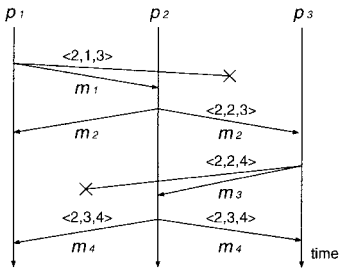

Figure 3: Suspected process.

4 Concluding Remarks

In this paper, we discussed how to detect faulty processes in a group in presence of disconnection of channel. We are now discussing how to make a consensus on a some view among all the operational processes.

References

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

[2] 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.

[3] Reiter, M. K., "The Rampart Toolkit for Building High-Integrity Services," *Theory and Practice in Distributed Systems, LNCS 938*, Springer-Verlag, 1995, pp.99–110.

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