2010 International Conference on Complex, Intelligent and Software Intensive Systems

Reduction of Messages Unnecessarily Ordered in Scalable Group Communication

Isamu Tsuneizumi
Seikei University
tsuneizumi.isamu873@gmail.com

Ailixier Aikebaier
Seikei University
alisher.akber@computer.org

Tomoya Enokido
Risho University
eno@ris.ac.jp

Makoto Takizawa
Seikei University
makoto.takizawa@computer.org

Abstract

In distributed peer-to-peer (P2P) applications, a group of multiple peer processes (peers) are required to cooperate with each other. Messages sent by peers have to be causally delivered. In this paper, we discuss a scalable group communication protocol for a group of multiple peers in P2P overlay networks. Due to the message overhead $O(n)$ for the number n of peers, the vector clock cannot be used to causally deliver messages is a scalable group. On the other hand, the linear clock implies the message length $O(1)$, but some pair of messages are unnecessarily ordered. Recently, more accurate physical clocks can be used with the GPS time server and network time protocol (NTP). In this paper, we consider a group where every member peer can use a physical clock which is synchronized with the time server in the network time protocol (NTP). Even if each physical clock is synchronized with a time server, every physical clock does not show the same accurate time. The accuracy of the physical clock depends on distance to the time server, traffic in a network and operating system. In this paper, we reduce the number of messages unnecessarily ordered by taking advantage of the linear time and physical time.

1. Introduction

Information systems based on the Peer-to-Peer (P2P) models [13] are now used and developed in various types of application. Here, a group of multiple peer processes (peers) are cooperating to achieve some objectives in a distributed network because a P2P system is in nature fully distributed without centralized coordinator. A group of multiple peers are cooperating by exchanging messages with each other in P2P overlay networks. Here, messages are required to be delivered to peers in some order like causally precedent order [11] and total order [1]. In this paper, we discuss a fully distributed group communication protocol for a scalable P2P group. The vector clock is widely used to causally deliver messages in group comment protocols [1, 4]. Only and all the messages to be causally ordered can be ordered in the vector clock. However, the vector clock [6] cannot be used in P2P networks due to the message overhead $O(n)$ for the number n of peers in scalable groups. In addition, it is not easy to change the membership of the group in every peer.

We assume a group is composed of multiple peers with GPS time servers. Physical clock of each peer is synchronized with a GPS time server [3] by using the network time protocol (NTP) [7]. Each peer is assumed to have a physical clock whose minimum accuracy is bounded to be some value because the peer is synchronized with the time server in a high-speed local network. The accuracy of each physical clock used by a peer depends on the distance and traffic between the peer and the time server. Hence, each physical clock of a peer is less accurate than the time server due to the delay time and traffic in the network.

In this paper, we consider a homogeneous broadcast group, where delay time between every pair of peers is the same and the accuracy of every physical clock is the same. In addition, each peer sends a message to every other peer in a group. Messages can be delivered to peers in the causal order with the linear clock [5]. However, some messages ordered with the linear clock might not be required to be causally ordered. In the physical clock, messages can be delivered to peers with not only the causal order but also the total order. More number of messages which are not required to be causally ordered are ordered in the physical clock. Thus, some message are *unnecessarily ordered* in the linear clock and physical clock. We discuss how messages can be reduced to be unnecessarily ordered by taking advantage of the physical and linear types of clocks depending on the accuracy of physical clock and the delay time

978-0-7695-3967-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CISIS.2010.119

among peers.

In section 2, we present a system model. In section 3, we present the precedent relation of messages. In section 4, we discuss logical clocks and physical clocks. In section 4, we discuss how to order messages in a homogeneous group. In section 5, we discuss the data transmission procedure in group protocol.

2 System Model

A group G is composed of multiple peers $p_1, \cdots, p_n$ which are interconnected in underlying communication networks. We assume the underlying networks are synchronous. The maximum delay time δ_{ij} between every pair of peers p_i and p_j is bounded to be some value. The networks are in addition reliable and high-speed ones. Even if the networks themselves are reliable, a peer might lose messages due to buffer overrun.

In this paper, we consider a fully distributed group G of multiple peers. Here, there is no centralized coordinator. Each peer directly communicates with other peers. A group is so scalable that a huge number of peers are included. In this paper, we assume there is some mechanism to broadcast a message to every member peer in a group by using the P2P overlay networks. In addition, each peer is autonomous and makes a decision on the delivery order of messages and the membership by itself.

In this paper, we assume a group G to include one or more than one GPS time server. Each peer p_i is assumed to be able to communicate with a GPS time server in a high-speed communication network e.g. a GPS time server in a some local area network as p_i. Each peer p_i is equipped with a physical clock c_i. A physical clock c_i of each peer p_i is synchronized with the GPS clock [3] of the time server in the network time protocol (NTP) [7]. We assume each pair of GPS clocks in a group G show the same accurate time. In a group, while every physical clock is synchronized with the GPS clock in NTP, each physical clock does not show the same accurate time depending on the delay time and traffic of the network with the time server.

Let PT_i show physical time shown by the physical clock c_i of a peer p_i. $C(PT_i)$ shows the accurate time when the physical clock c_i of a peer p_i shows time PT_i. The difference the physical clock c_i of between the physical time and accurate time $\mid C(PT_i) - PT_i \mid$ gives the *accuracy* of the physical clock of a peer p_i. The accuracy of the physical clock c_i of each peer p_i is assumed to be bounded to be some constant value λ_i. We assume the following condition holds for every peer p_i:

$$|C(PT_i) - PT_i| \leq \lambda_i \text{ for every physical time } PT \quad (1)$$

For example, let us consider a system where every computer is connected in a Gigabit Ethernet LAN and a physical clock of every computer is synchronized in NTP. Here, λ_i is 1 to 10 [msec] according to our experiment [12]. λ_i is about 100 to 500 [msec] in a system where computers are interconnected in a wide area network (WAN). The longer and more complicated the network is, the less accurate the physical clock. In fact, λ_i move depends on a type of operating system, which a peer p_i is performed. Unit types of operating systems shows more accurate time than Windows [12]. In the physical clock condition (1), λ_i depends on distance and traffic between a peer p_i and a time server. Since each peer p_i is synchronized with a time server in a high-speed local network, λ_i is considered to be the order of millisecond.

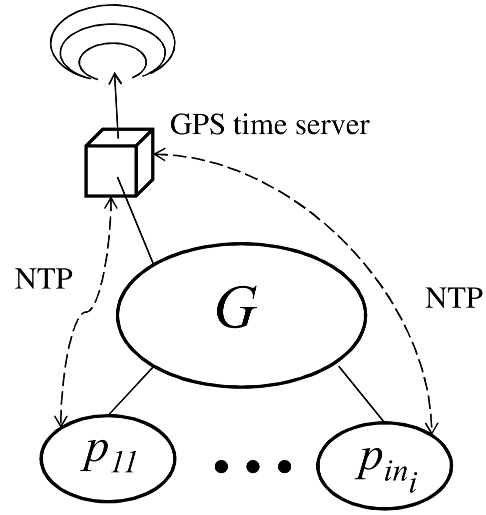

Figure 1. Group

There are two types of groups with respect to which peer a message is sent in a group. Each peer sends a message to every peer in a *broadcast* group. A message which a peer receives is also received by every other peers. On the other hand, in a *selective broadcast* group, a peer may send a message to a subset of the peers. If a pair of peers p_i and p_j send messages m_i and m_j, respectively, only common destinations of m_i and m_j receive both the messages m_i and m_j. In this paper, we consider a *broadcast* group. We assume there is some mechanism where each peer can broadcast a message to every peer in a group. There are father types of groups, homogeneous and heterogeneous groups. In a homogeneous group, the accuracy λ_i of physical clock c_i of every peer p_i is the same, $\lambda_i = \lambda$ and the delay time $\delta_{ij} = \delta$. Typically, computers of the same type are interconnected in a high-speed local area network. In heterogeneous group, $\lambda_i \neq \lambda$ for peers p_i, p_j, p_k, and p_l. The example is a group where various types of computer are interconnected in various type of networks. In this paper, we consider a broadcast and homogeneous type of a scalable group.

3 Precedent Relations of Messages

In a group G of peers $p_1, \cdots, p_n$ ($n \geq 1$), messages sent by peers have to be delivered to destination peers in the causally precedent order [5, 4]. Let $s_i[m]$ and $r_i[m]$ denote sending and receipt events that a peer p_i sends and receives a message m, respectively, in the group G. Lamport [5] defines the famous *happen-before* relation $e_1 \rightarrow e_2$ between a pair of events e_1 and e_2 (e_1 *happens before* e_2) in a distributed system. $e_1 \Rightarrow e_2$ shows that e_1 physically happens before e_2. For a pair of messages m_1 and m_2, two types of precedent relations among the messages m_1 and m_2 are defined in terms of the happens-before relations $\rightarrow$ and $\Rightarrow$ as follows:

1. m_1 *causally precedes* m_2 ($m_1 \rightarrow m_2$) if and only if (iff) $s_i[m_1] \rightarrow s_j[m_2]$ [5] [Figure 2].
2. m_1 *temporally precedes* m_2 ($m_1 \Rightarrow m_2$) iff $s_i[m_1] \Rightarrow s_j[m_2]$ [Figure 3].

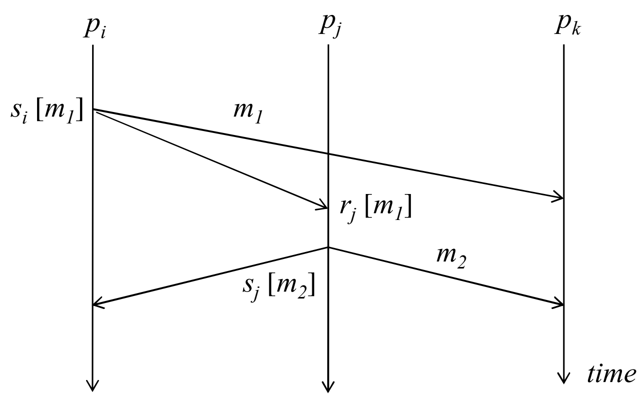

Figure 2. Causality ($m_1 \rightarrow m_2$).

From the definitions, a message m_1 temporally precedes a message m_2 ($m_1 \Rightarrow m_2$) if m_1 causally precedes m_2 ($m_1 \rightarrow m_2$). However, even if $m_1 \Rightarrow m_2$, $m_1 \rightarrow m_2$ does not necessarily hold. A pair of messages m_1 and m_2 are referred to as *causally concurrent* ($m_1 \mid m_2$) iff neither $m_1 \rightarrow m_2$ nor $m_2 \rightarrow m_1$. Causally concurrent messages m_1 and m_2 are independent of one another. Each peer can receive m_1 and m_2 in any order. A pair of messages m_1 and m_2 are referred to as *temporally concurrent* ($m_1 \parallel m_2$) iff neither $m_1 \Rightarrow m_2$ nor $m_2 \Rightarrow m_1$. $m_1 \parallel m_2$ means that a pair of messages m_1 and m_2 are sent at the same time. Here, $m_1 \mid m_2$ if $m_1 \parallel m_2$.

For example, there is a group of three peers p_1, p_2, and p_3. Suppose the peer p_1 sends a question message Q_1 and p_2 sends an answer message A_1 of the question Q_1 in a teleconference application. Here, Q_1 causally precedes A_1 ($Q_1 \rightarrow A_1$) since the answer A_1 is meaningless without the question Q_1. Every peer p_k is required to deliver the question message Q_1 and then the answer message A_1. On the other hand, a pair of the peers p_1 and p_2 send question messages Q_1 and Q_2, respectively. If Q_1 is sent before Q_2, Q_1 temporally precedes Q_2 ($Q_1 \Rightarrow Q_2$). Then, the peer p_3 sends an answer message A_1 of Q_1 and then A_2 of Q_2.

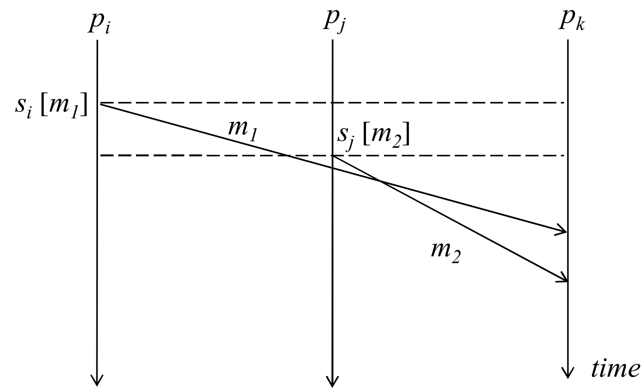

Figure 3. Temporal precedence ($m_1 \Rightarrow m_2$).

4 Types of Clocks

4.1 Logical Clocks

In order to deliver messages in the causally precedent order, synchronization mechanisms based on types of logical clocks; linear clock [5] and vector clock [8, 6] are used in traditional group communication protocols [11, 10]. In the linear clock, each peer p_i has a variable LT_i showing the linear time of its linear clock. Each time a peer p_i sends a message m, the message m carries the linear time $m.LT$ shown by the linear clock LT_i. Then, the peer p_i sends the message m and LT_i is incremented by one. On receipt of a message m, the peer p_i changes the variable LT_i with the maximum value of LT_i and $m.LT$, i.e. LT_i = max (LT_i, $m.LT$). Here, if the message m_1 causally precedes the message m_2 ($m_1 \rightarrow m_2$), $m_1.LT < m_2.LT$. Since messages which are causally concurrent are ordered, some messages may be unnecessarily delayed to be delivered. However, $m_1 \rightarrow m_2$ may not hold even if $m_1.LT < m_2.LT$. In the linear clock, m_1 is unnecessarily ordered to precede m_2 ($m_1 \prec m_2$) if $m_1.LT < m_2.LT$. The message size is $O(1)$ independent of the number of peers.

In the vector clock [8, 6], each peer p_i has a vector $\langle VT_1, \cdots, VT_n \rangle$ of logical time for a group of peers $p_1, \cdots, p_n$ ($n > 1$). Suppose a peer p_i would like to send a message m.

First, the vector $VT = \langle VT_1, \cdots, VT_n \rangle$ is carried by the message m, $m.VT = \langle VT_1, \cdots, VT_n \rangle$. Then, the peer p_i sends the message m and the ith element VT_i of the vector dots VT is incremented by one. On receipt of a message m, $VT_j = max(VT_j, m.VT_j)$ for $j = 1, \cdots, n$ ($j \neq i$) in the peer p_i. A message m_1 causally precedes another message m_2 ($m_1 \to m_2$) if and only if (iff) $m_1.VT < m_2.VT$. Only and all the messages to be causally ordered are ordered in the vector clock. However, the message size is $O(n)$ for the number n of peers since each message m carries the vector $m.VT$ of n elements. The vector clock cannot be adopted in scalable group communication protocols due to the communication overhead $O(n)$. In addition, it is not easy to change the vector in change of the membership of the group.

In the group communication protocols [4, 5, 7], the underlying networks are assumed to be reliable. In fact, a peer may lose messages due to buffer overrun even if the network is reliable and high-speed one. In the paper [11], every message can be causally delivered by taking usage of sequence numbers of messages even if some messages are lost in a network.

4.2 Physical clocks

A physical clock of each peer can now show more accurate time by obtaining time from a time server with GPS clock [3] by using the network time protocol (NTP) [7]. Each peer p_i has a variable PT_i showing the current time of the physical clock c_i. The physical clock c_i is synchronized with the time server in the group G. A peer p_i attaches a message m with the physical time $m.PT$. Let $m.PT$ show the physical time PT_i of a source peer p_i when p_i sends a message m. Then the peer p_i sends the message m in the group G. Here, the message length is $O(1)$ as the linear clock.

Let $C_i(PT_i)$ show the accurate time when the physical clock of a peer p_i shows the physical time PT_i. As assumed in the preceding subsection, $|PT_i - C(PT_i)| \leq \lambda_i$ for every time PT_i in every peer p_i. For every pair of peers p_i and p_j, $|C_i(PT) - C_j(PT)|$ show time difference between the physical clocks of p_i and p_j when $PT_i = PT$ and $PT_j = PT$. Hence, $|C_i(PT) - C_j(PT)| \leq \lambda_i + \lambda_j$ for every physical time PT to be shown by a pair of peers p_i and p_j.

It takes time for a peer to transmit a message to another peer. Suppose a peer p_i sends a message m at physical time PT_i and another peer p_j receives the message m at physical time PT_j as shown in Figure 4. Here, $|C_i(PT_i) - C_j(PT_j)|$ shows the delay time between a pair of peers p_i and p_j. In this paper, we assume the network is synchronous [2], i.e. the maximum delay time between a pair of peers p_i and p_j is bounded to be a constant δ_{ij}. Here, the following property holds for physical time PT_i and PT_j when a peer p_i sends the message m and another peer p_j receives the message m, respectively, clock accuracy λ_i and λ_j, and delay time δ_{ij}:

$$-(\lambda_i + \lambda_j) + \delta_{ij} \leq PT_j - PT_i \leq (\lambda_i + \lambda_j) + \delta_{ij}. \quad (2)$$

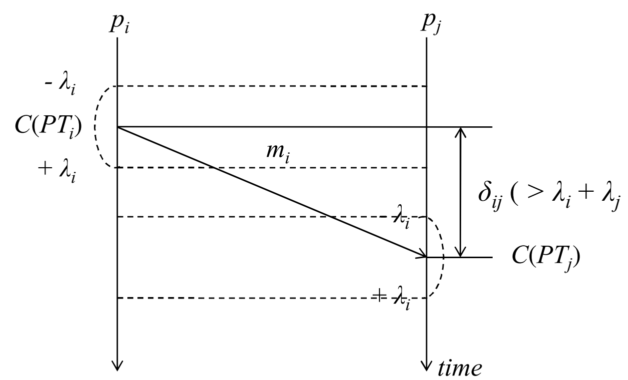

Figure 4. Accuracy and delay time.

Next, suppose a peer p_i sends a message m_i and another peer p_j sends a message m_j. First, if $m_j.PT - m_i.PT > \lambda_i + \lambda_j$, it is sure that a message m_i is sent by a peer p_i before another message m_j by a peer p_j, i.e. m_i temporally precedes m_j ($m_i \Rightarrow m_j$). If $m_j.PT - m_i.PT > \lambda_i + \lambda_j + \delta_{ij}$, it is sure that the peer p_j sends the message m_j after receiving the message m_i, i.e. m_i causally precedes m_j ($m_i \to m_j$). Thus, the following properties hold:

$$\text{If } m_j.PT - m_i.PT > \lambda_i + \lambda_j, \quad s_i[m_i] \Rightarrow s_j[m_j]. \quad (3)$$

$$\text{If } m_j.PT - m_i.PT > \lambda_i + \lambda_j + \delta_{ij}, \quad s_j[m_i] \to s_j[m_j]. \quad (4)$$

Suppose a peer p_i sends a messages m_i at physical time PT_i and another peer p_j receives the message m_i at physical time PT_j. Suppose the peer p_j sends a message m_j. Here, $-(\lambda_i + \lambda_j) + \delta_{ij} \leq PT_j - m_i.PT \leq \lambda_i + \lambda_j + \delta_{ij}$ holds from the property (2). Hence, if $m_j.PT - m_i.PT > \lambda_i + \lambda_j + \delta_{ij}$, it is sure $m_j.PT > PT_j$. This means the peer p_j sends the message m_j after receiving the message m_i from the peer p_i, i.e. m_i causally precedes m_j ($m_i \to m_j$).

Let $e.PT$ show the physical time of a peer where an event e occurs. Here, we define the temporally precedent relation $\Rightarrow$ among a pair of events e_i and e_j occurring at peers p_i and p_j, respectively:

[Definition] An event e_i *temporally precedes* an event e_j ($e_i \Rightarrow e_j$) iff $e_j.PT - e_i.PT > \lambda_i + \lambda_j$.

A pair of events e_i and e_j are *temporally concurrent* ($e_i \parallel e_j$) if neither $e_i \Rightarrow e_j$ nor $e_j \Rightarrow e_i$, i.e. $|e_j.PT - e_i.PT| \leq \lambda_i + \lambda_j$. Hence, $s_i[m_i] \Rightarrow s_j[m_j]$ iff $m_j.PT - m_i.PT > \lambda_i + \lambda_j$. A pair of sending events $s_i[m_i]$ and $s_j[m_j]$ concurrently occur ($s_i[m_i] \parallel s_j[m_j]$), i.e. a pair of peers m_i and m_j are sent at the same time iff $|m_j.PT - m_i.PT| \leq \lambda_i + \lambda_j$.

Suppose another peer p_k receives a pair of messages m_i and m_j sent by peers p_i and p_j, respectively. The peer p_k has to decide on the precedence of the messages m_i and m_j by using the physical time $m_i.PT$ and $m_j.PT$ carried in the messages m_i and m_j, respectively. Following the properties, the temporally precedent relation $m_i \Rightarrow m_j$ and the temporally concurrent relation $m_i \parallel m_j$ on a pair of messages m_i and m_j satisfy the following properties:

$$m_i \Rightarrow m_j \text{ if } m_j.PT - m_i.PT > \lambda_i + \lambda_j. \quad (5)$$

$$m_i \parallel m_j \text{ if } |m_j.PT - m_i.PT| \leq \lambda_i + \lambda_j. \quad (6)$$

Suppose a peer p_j sends a message m_j after receiving a message m_i from another peer p_i, i.e. m_i causally precedes m_j ($m_i \rightarrow m_j$). Here, $m_j.LT > m_i.LT$ since $m_i \rightarrow m_j$ in the linear clock. However, even if $|\, m_j.PT - m_i.PT \,| < \lambda_i + \lambda_j + \delta_{ij}$, $m_j.PT > m_i.PT$ may not hold. The physical time PT if m_j might be larger than m_i although $m_i \rightarrow m_j$. Next, suppose a peer p_j sends a message m_j after receiving a message m_i from a peer p_i. Here, the maximum delay time between a pair of the peers p_i and p_j is δ_{ij}. The, following property holds from the property (2):

$$\text{A peer } p_j \text{ sends a message } m_j \text{ after receiving a message } m_i \text{ from a peer } p_i \text{ if } |m_j.PT - m_i.PT| > \lambda_i + \lambda_j + \delta_{ij}. \quad (7)$$

We would like to discuss how the physical clock can be used to causally order messages with respect to the delay time δ_{ij} and the clock accuracy λ_i and λ_j. First, suppose $\lambda_i + \lambda_j < \delta_{ij}$. That is, the delay time between a pair of peers is longer and the accuracy of a physical clock is better. For example, peers are interconnected in a wide area network. As presented in the condition (1), $m_j.PT - \lambda_j \leq C(m_j.PT) \leq m_j.PT + \lambda_j$. If $C(m_j.PT) > C(m_i.PT) + \delta_{ij}$, it is sure the peer p_j sends the message m_j after receiving the message m_i from the peer p_i. Hence, $m_j.PT + \lambda_j \geq C(m_j.PT) > C(m_i.PT) + \delta_{ij} \geq m_i.PT - \lambda_i + \delta_{ij}$. $m_j.PT > m_i.PT - (\lambda_i + \lambda_j) + \delta_{ij}$. If $\delta_{ij} \geq \lambda_i + \lambda_j$, $m_j.PT > m_i.PT - (\lambda_i + \lambda_j) + \delta_{ij} \geq m_i.PT$. Thus, if $\delta_{ij} \geq \lambda_i + \lambda_j$ and m_i causally precedes m_j ($m_i \rightarrow m_j$), $m_j.PT - m_i.PT > 0$. Hence, by checking the physical time $m_i.PT$ and $m_j.PT$ of every pair of messages m_i and m_j, we can decide on which message m_i or m_j causally precedes another message.

[Theorem] If $\delta_{ij} \geq \lambda_i + \lambda_j$ and $m_j.PT - m_i.PT > \lambda_i + \lambda_j + \delta_{ij}$, a message m_i causally precedes a message m_j ($m_i \rightarrow m_j$).

Next, suppose $\delta_{ij} < \lambda_i + \lambda_j$. The condition $\delta_{ij} < \lambda_i + \lambda_j$ implies that the network is a high-speed network and each physical clock is not accurate. Here, $m_j.PT < m_i.PT$ may hold even if $m_i \rightarrow m_j$. Hence, only by using the physical clock, we cannot decide on the causality of every pair of messages. The messages can be causally ordered in the linear clock. That is, if $m_j.LT > m_i.LT$, m_i is ordered prior to m_j ($m_i \mapsto m_j$).

5 Homogeneous Broadcast Groups

There are types of groups with respect to the delay time and clock accuracy. In this paper, we consider a *homogeneous broadcast* group $G = \{ p_1, \cdots, p_n \}$ ($n \geq 1$) where $\delta_{ij} = \delta$ and $\lambda_i = \lambda$ for every pair of peers p_i and p_j. In the homogeneous group G, the delay time δ_{ij} between a pair of peers p_i and p_j is the same δ and the accuracy λ_i of every physical clock of a peer p_i is also the same λ. Each message m carries two types of time, physical time $m.PT$ and linear time $m.LT$ which show the physical time and linear time of the source peer, respectively.

Let us consider a pair of messages m_i and m_j sent by peers p_i and p_j, respectively. Suppose another peer p_k receives the messages m_i and m_j. First, suppose $|m_i.PT - m_j.PT| < 2\lambda$. That is, we cannot decide which message m_i or m_j is sent before the other message by comparing the physical time $m_i.PT$ and $m_j.PT$, i.e. m_i and m_j are temporally concurrent ($m_i \parallel m_j$).

First, suppose the maximum delay time δ between a pair of peers is smaller than 2λ, $\delta < 2\lambda$ as shown in Figure 5. Hence, a peer p_i sends a message m_i at physical time PT_i and another peer p_j receives the message m_i at PT_j. Here, $PT_i + \lambda > PT_j - \lambda$ may hold.

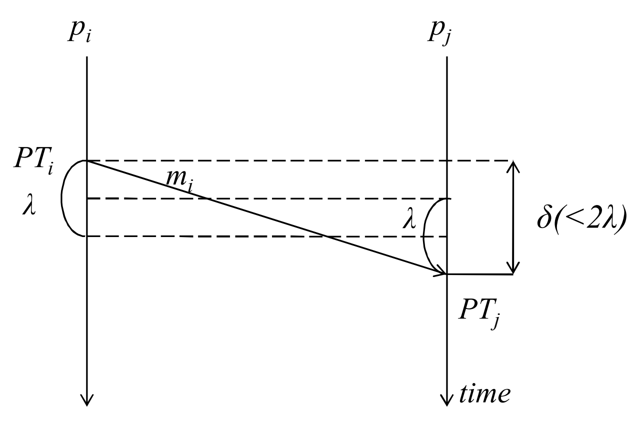

Figure 5. Delay time.

As discussed in the preceding section, we cannot decide on which causally precedent relation $m_i \rightarrow m_j$ or $m_j \rightarrow m_i$ holds if $\delta < 2\lambda$ only by using the physical time $m_i.PT$ and $m_j.PT$. We need an additional mechanism, i.e. linear clock to order messages. If $m_i.LT < m_j.LT$, the message m_i is ordered to precede the message m_j ($m_i \prec m_j$). However, $m_i \rightarrow m_j$ may not hold. Thus, m_i can be ordered to precede m_j ($m_i \prec m_j$) by using the linear time even if ($m_i \nrightarrow m_j$). If $\delta < 2\lambda$, messages have to be ordered by using the physical clock and the linear clock.

Next, suppose $\delta \geq 2\lambda$. As shown in a Figure 6, a peer p_i sends a message m_i at physical time PT_i and another peer p_j receives the message m_i at physical time PT_j. Here, $PT_i + \lambda < PT_j - \lambda$. Hence, we can decide on whether or not the messages m_i and m_j are causally ordered by using the physical time $m_i.PT$ and $m_j.PT$. As presented in the property (7), if m_i causally precedes m_j ($m_i \rightarrow m_j$), $m_j.PT - m_i.PT > \delta + 2\lambda$. That is, if $m_j.LT > m_i.LT$, $m_j.PT - m_i.PT > \delta + 2\lambda$. Hence, if $m_j.PT - m_i.PT > \delta + 2\lambda$, the message m_i causally precede m_j ($m_i \rightarrow m_j$) and m_i can be ordered to precede m_j ($m_i \mapsto m_j$). This means, messages can be causally ordered by using only the physical time if $\delta \geq 2\lambda$.

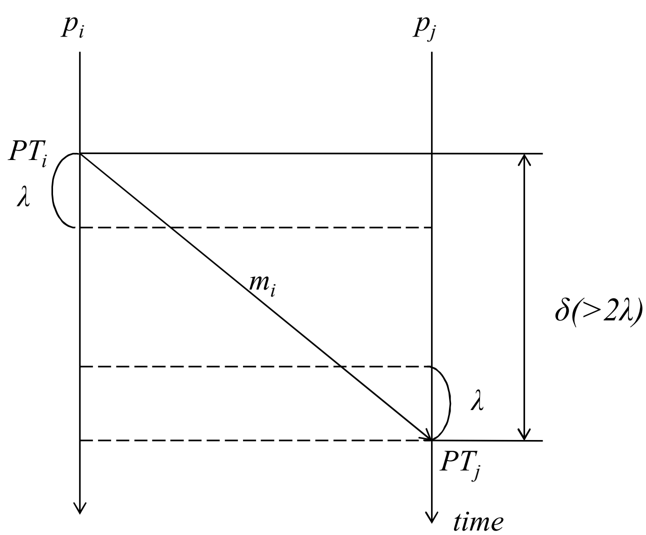

Figure 6. $\delta > 2\lambda$.

Next, suppose $|m_i.PT - m_j.PT| > 2\lambda$. Here, if $m_j.PT > m_i.PT$, it is sure a message m_i temporally precedes another message m_j ($m_i \Rightarrow m_j$). As discussed here, messages are causally ordered by using the physical / linear clock depending on whether or not $\delta > 2\lambda$ holds.

In summary, messages are temporally ordered by using the following rules:

[**Temporally ordring rules**]

1. If $|m_i.PT - m_j.PT| \leq 2\lambda$, a pair of messages m_i and m_j are temporally concurrent ($m_i \parallel m_j$).
2. If $m_j.PT - m_i.PT > 2\lambda$, a message m_i temporally precedes a message m_j ($m_i \Rightarrow m_j$).

We discuss how to order a pair of messages m_i and m_j by using the physical time and linear time. Suppose a peer p_k receives a pair of messages m_i and m_j from peers p_i and p_j, respectively. The messages are ordered in the precedent relation $\mapsto$ by the following rules. Here, a pair of messages m_1 and m_2 are ordered as "m_1 precedes m_2" ($m_1 \mapsto m_2$), ($m_2 \mapsto m_1$), or "m_1 and m_2 are concurrent" ($m_1 \leftrightarrow m_2$).
[**Rules for ordering messages** m_i **and** m_j]

1. $|\ m_i.PT - m_j.PT\ | \leq 2\lambda + \delta$:
   (a) $\delta \leq 2\lambda$:
   if $m_i.LT < m_j.LT$, m_i precedes m_j ($m_i \mapsto m_j$)
   else if $m_j.LT < m_i.LT$, $m_j \mapsto m_i$
   else if $|\ m_i.PT - m_j.PT\ | \leq 2\lambda$, $m_i \leftrightarrow m_j$
   else neither $m_i \leftrightarrow m_j$.
   (b) $\delta > 2\lambda$:
   if $|\ m_i.PT - m_j.PT\ | \leq 2\lambda$, $m_i \leftrightarrow m_j$
   else /* $|\ m_i.PT - m_j.PT\ | \geq 2\lambda$ */
   else if $m_i.PT < m_j.PT$, {
   if $m_i.LT < m_j.LT$, $m_i \mapsto m_j$
   else $m_i \leftrightarrow m_j$ }
   else /* $m_i.PT > m_j.PT$ */ {
   $m_j \Rightarrow m_i$
   if $m_i.LT > m_j.LT$, $m_j \mapsto m_i$
   else $m_i \leftrightarrow m_j$.
   }
2. $|\ m_i.PT - m_j.PT\ | > 2\lambda + \delta$;
   if $m_i.PT < m_j.PT$, $m_i \mapsto m_j$ ($m_i \rightarrow m_j$)
   else $m_j \mapsto m_i$ ($m_j \rightarrow m_i$).

In the rule 2, $m_i \mapsto m_j$ means $m_i \rightarrow m_j$. Figure 7 shows pairs of messages ordered in the precedent relations $\Rightarrow$, $\mapsto$, $\rightarrow$ and linear clock LT. For a pair of messages m_i and m_j, $m_i \rightarrow m_j$ if $m_i \mapsto m_j$. $m_i \mapsto m_j$ if $m_i \Rightarrow m_j$. In the linear clock, a pair of message m_i and m_j are ordered as $m_i \prec m_j$ if $m_i.LT < m_j.LT$. However, even if $m_i.LT < m_j.LT$, $m_i \rightarrow m_j$ may not hold. In our protocol, $m_i \mapsto m_j$ if $m_i.PT < m_j.PT$ and $|\ m_i.PT - m_j.PT\ | > 2\lambda + \delta$. Here, we can reduce the number of messages to be ordered.

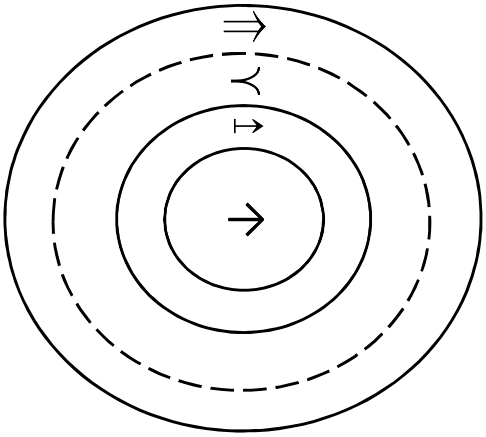

Figure 7. Relation among precedent relations

6 Data Transmission Procedure

We present a data transmission procedure of a homogeneous broadcast group $G = \{p_1, \cdots, p_n\}$. First, we assume the underlying network to be synchronous [2]. A peer p_i broadcasts a message m to every peer by using a primitive function $\mathbf{send}(m)$ in the underlying P2P overlay network. A peer receives a message m by using a primitive function m = $\mathbf{rec}()$.

Each message m carries the following fields:

$m.src$ = source peer which sends a message m.
$m.seq$ = sequence number of the message m.
$m.PT$ = physical time.
$m.LT$ = linear time.
$m.data$ = data.

Each peer p_i manipulates variables LT_i, PT_i, SEQ_i, and REQ_j (j = 1, $\cdots$, n) to send and receive messages. PT_i shows the current physical time of the peer p_i. LT_i indicates the linear time of p_i. Initially, LT_i = 0. SEQ_i shows the sequence number of a message which the peer p_i has most recently sent in the network. Initially, SEQ_i = 0. REQ_j indicates the sequence number of a message which a peer p_i expects to receive next from p_j, initially REQ_j = 0 (j = 1, $\cdots$, n).

First, a peer p_i sends a message m of data $DATA$ by the following procedure:

[**Transmission**]
  $m.src = p_i$;
  $m.LT = LT_i$;
  $m.PT = PT_i$;
  $m.data = DATA$;
  $m.seq = SEQ_i$;
  $\mathbf{send}(m)$;
  $SEQ_i = SEQ_i + 1$;
  $LT_i = LT_i + 1$;

Next, a peer p_i receives a message m from another peer p_j as follows:

[**Receipt**]
  m = **rec** (); /* m comes from p_j , i.e. $m.src = p_j$ */
  if $REQ_j < m.seq$,
  {
    /* there is a lost message m' such that
    $REQ_j \leq m'.seq < m.seq$ */
   recovery ($p_j, REQ_j, m.seq$ - 1); /* lost messages
    are retransmitted */
  }
  else if $REQ_j > m.seq$,
  {
    /* m is already received */ **neglect** m;
  }
  else /* $REQ_j = m.seq$ */
  {
    LT_i = max(LT_i,$m.LT$);
    enque(m, RBF); /* buffer m to RBF */
    $REQ_j = REQ_j + 1$;
  }

If $REQ_j < m.seq$, a peer p_i finds p_i has not received a message m' from a peer p_j where $m'.seq \geq REQ_j$ and $< m.seq$. If a peer p_i detects to lose the message m', p_i requests the source peer p_j to retransmit m'. In the function **recovery**(p_j, REQ_j, $m.seq$ - 1), the source peer p_j retransmits the peer p_i every message m' where $REQ_j \leq m'.seq < m.seq$ - 1, i.e. every message which p_i has lost.

Messages received are enqueued into the receipt buffer RBF. Here, the messages in the buffer RBF are ordered by using the ordering rules discussed in the preceding section. Here, a function m = **top**(RBF) shows that a message m is the top of the buffer RBF. A function **enque**(m, RBF) means that a message m is stored in the buffer RBF and ordered in the ordering rule as presented in the preceding section. A function m = **dequeue**(RBF) shows that the top message **top**(RBF) is dequeued from the buffer RBF. The top message m in the buffer RBF may not be delivered to the peer p_i. If the message m satisfies the following delivery condition **DelCond**(m), the peer p_i can deliver the message m. Otherwise, the top message m is kept waited in the buffer RBF. Here, a variable $DSEQ_j$ shows the sequence number of a message from a peer p_j which the peer p_i has most recently delivered (j = 1, $\cdots$, n).

[**DelCond**(m)]
  {
   /* m comes from p_j, i.e. $p_j = m.src$ */
   if $DSEQ_j = m.seq - 1$,
    if there is a message m_k in RBF such that
    $DSEQ_k = m_k.seq$ - 1 for every peer p_k,
    return (T);

return (F);
}

This condition means that every message preceding the message m is delivered. Hence, the message m which is the top of the buffer RBF can be delivered. Otherwise, after delivering the message m, some message preceding the message m might be received.

A peer p_i delivers messages in the buffer RBF by the following **Deliver** procedure:

[**Deliver**]
```
m = top(RBF);
if DelCond(m), /* m can be delivered */
{
    m = deque (RBF);
    deliver (m); /* m is delivered */
    DSEQ_j = m.seq + 1; /* p_j = m.src */
}
```

7 Concluding Remarks

In this paper, we discussed a homogeneous broadcast group of multiple peers where the delay time between every pair of peers is the same and the clock accuracy of every peer is the same. Here, each message is sent to every peer in the group. Since the linear clock implies the message length $O(1)$, we can adopt the linear clock to a scalable group. However, some messages are unnecessarily ordered. In order to reduce the number of messages to be unnecessarily ordered, we take advantage of the physical clock in addition to the linear clock. We presented the data transmission procedure of the group communication protocol.

Acknowledgment

This research was partially supported by the strategy researched projet of Seikei university and MEXT, Grant-in-Aid for Building Strategy Research Infrastructure.

References

[1] K. P. Birman and R. Van Renesse : Reliable Distributed Computing with the Isis Toolkit (Systems). *Wiley-IEEE Computer Society publisher*, ISBN-0818653426, 1994.

[2] M. J. Fischer, N. A. Lynch and M. S. Paterson : Impossibility of Distributed Consensus with One Faulty Process. *In: proc. ACM publisher*, volume 32, pages.374-382, 1985.

[3] B. Hofmann-Wellenhof, H. Lichtenegger and J. Collins : Global Positioning System (GPS). Theory and practice. *SpringerWienNewYork*

[4] M. F. Kaashoek and A. S. Tanenbaum : Group Communication in the Amoeba Distributed Operating System. *In: Proc. of the IEEE 11th International Conference on Distributed Computing Systems*, pages.222-230, 1991.

[5] L. Lamport : Time, Clocks, and the Ordering of Events in a Distributed System. *In: proc. of the Communication of the ACM,* volume 21, pages.558-565, 1978.

[6] F. Mattern : Virtual Time and Global States of Distributed Systems. *North-Holland publisher* pages.215-226, 1989.

[7] D. L. Mills : Network Time Protocol (NTP). *RFC Editor publisher* 1985.

[8] F. Mattern : Algorithms for Distributed Termination Detection. *In: proc. of the Distributed Computing journal,Springer Berlin publisher,* volume 2, number 3, pages.161-175, 1987.

[9] N. Mark : Gigabit Ethernet Technology And Applications. *Artech Hou publisher*, ISBN-1580535054.

[10] L. E. Moser, P. M. Melliar-Smith, and V. Agrawala : Membership Algorithms for Asynchronous Distributed Systems. *In: proc. of the 10th International Conference on Distributed Computing Systems*, pages.480-488, 1991.

[11] A. Nakamura and M. Takizawa : Causally Ordering Broadcast Protocol. *In: proc. of the IEEE 14th International Conference on Distributed Computing Systems*, pages.48-55, 1994.

[12] T. Nishimura, N. Hayashibara, M. Takizawa, and T. Enokido : Causally Ordered Delivery with Global Clock in Hierarchical Group. *In: proc. of the IEEE 11th International Conference on Parallel and Distributed Systems*, pages.560-564, 2005.

[13] R. Schollmeier : A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. *In: proc. of the First International Conference on Peer-to-Peer Computing*, volume 0, pages.101-102, 2001.