社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－DPS－113　(16)
2003／6／20

Communication Protocol for Heterogeneous Group

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

Abstract

Information systems are composed of various types of networks like personal area networks, local area networks, and wide-area networks. Processes distributed in these types of networks are autonomically cooperating to achieve some objectives in peer-to-peer (P2P) applications. Processes have to support physical or logical clocks in order to synchronize the processes, e.g. to causally ordered messages. We discuss group communication protocol named HCG (heterogeneous clock group) where a group is composed of subgroups which are interconnected with the Internet. Processes in each subgroup are interconnected with local and personal area network and use physical or liner clocks. On the other hand, processes in the Internet use vector clock.

異種時計ネットワーク間におけるグループ通信プロトコル

河浪 悟士　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 {kawa, eno, taki}@takilab.k.dendai.ac.jp

現在の情報システムは、複数の計算機が PAN、LAN、WAN 等のネットワークにより相互接続された分散システムとなっている。アプリケーションは、各計算機上に分散された複数のプロセスがグループを構成し、グループ内のプロセスが互いにメッセージを交換し協調動作を行うことで実現される。このようなグループ内のプロセス間の通信プロトコルでは、メッセージを因果順序に配送する必要がある。メッセージの因果順序配送を保証する手法として、ベクタ時刻という論理時間、または、各計算機の物理時間を同期させる手法が提案されている。しかし大規模グループでは、ベクタ時刻の処理と通信の負荷が大きく、大規模グループ内のコンピュータ間では物理時計を同期させることが困難である。本論文では、大規模グループを構成するネットワーク範囲ごとに最適な論理・物理時間を適用し、メッセージの因果順序配送を行う方法を提案する。

1. Introduction

Distributed systems are composed of multiple processes interconnected with networks. Peer processes are cooperating to achieve some objectives by exchanging messages with each other in peer-to-peer (P2P) applications [10]. A collection of cooperating peer processes is referred to as *group*. Messages have to be causally delivered to processes in a group [4, 8]. There are many discussions on group communication protocols [1, 9, 15], where messages are causally delivered by using the vector clocks [6].

Processes are connected with various types of networks like personal area network [12], local area network (LAN) [11], and wide-area network (WAN) [11]. In a personal area network, processes in *last ten's meters* are interconnected with wireless communication channels like Bluetooth [2] and IEEE 802.11b [5]. Each type of network is characterized by quality of service (QoS), i.e. delay time, bandwidth, and packet loss ratio. In order to synchronize processes in a group, types of clocks like logical clocks [4, 6] and physical clock are used. Some mechanisms to synchronize physical clocks in multiple processes are discussed like NTP (Network Time Protocol) [7] which take usage of TCP/IP [13, 14]. Clock synchronization in a one-hop *ad hoc* network is also discussed [3]. All the events occurring in a distributed system can be totally ordered by time-stamping messages with the physical clock. Since the message length is $O(1)$, it is easy to design and implement algorithms for synchronizing processes. However, these algorithms are not applicable for a wide-area network due to long delay time among processes. In the liner clock, message length is $O(1)$ as well as physical clock. Events not to be ordered are ordered. The vector clock [6] can be used to synchronize processes in a wide-area network but message length is $O(n)$ for number n of processes in a group. The computation and communication overheads are too large to realize a group including a large number of processes while only and all messages to be causally ordered can be ordered. In this paper, we discuss a structured group which is composed of subgroups, each of which takes usage of its own type of clock to synchronize local processes. For example, processes in a personal area network adopt their own physical clocks while processes in a wide-area network use vector clocks. Thus, a *heterogeneous* group is composed of subgroups with different types of clocks. In this paper, we consider a system where a collection of processes are interconnected in a local network, i.e. local and personal area network, and the local networks are further interconnected in a wide-area network. In a local network, processes are synchronized by using physical clocks or liner clock.

In a wide-area network, processes are synchronized by using vector clocks. Even messages not to be causally ordered are ordered in physical clock and liner clock. In addition, even if messages are locally causally ordered in a local group, the messages may be causally concurrent in a group. In this paper, we reduce the number of messages to be unnecessarily ordered in a structured, heterogeneous group.

In section 2, we present a system model. In section 3, we briefly overview clock synchronization techniques. In section 4, we discuss how to synchronize clocks. In section 5, we evaluate the HCG protocol.

2. System Model

2.1 Group

Distributed systems are composed of multiple cooperating processes $p_1, ..., p_n$ distributed on computers interconnected with various types of networks by exchanging messages. A *group* means a collection of peer processes $p_1, ..., p_n$ which are cooperating by exchanging messages. Processes in a group are required to be synchronized. For example, messages are required to be causally delivered to processes in a group.

Let $s_i(m)$ and $r_i(m)$ denote events showing that a process p_i sends and receives a massage m, respectively. The *happen-before* relation is defined by Lamport [4]. A message m_1 *causally precedes* another massages m_2 ($m_1 \rightarrow m_2$) if and only if (iff) $s_i(m)$ *happens before* $r_i(m)$. Each process is required to deliver a massage m_1 before another message m_2 if m_1 *causally precedes* m_2. A message m_1 is *causally concurrent* with another message m_2 ($m_1 \parallel m_2$) iff neither $m_1 \rightarrow m_2$ nor $m_2 \rightarrow m_1$. A process can deliver a pair of *causally concurrent* messages in any order.

2.2 Heterogeneous clocks

Each computer is equipped with a physical clock. However, every pair of physical clocks in different computers do not always show same time. Each computer has to synchronize its physical clock with the other computers in order to do the cooperation. NTP (Network Time Protocol) [7] is used to synchronize physical clocks by using TCP/IP. Processes communicate with a time server to obtain the current time. It takes time to exchange messages between the computer and the time server. The computer calculates the delay time to the time server and then estimates current time. If delay time is long and variant, the process cannot obtain correct current time. Lai [3] discusses how to synchronize clocks in a one-hop *ad hoc* network like personal area network (PAN) where delay time is short. A message m carries time-stamp $m.T$ which shows physical time when m is sent. In a *liner clock*, each process p_i manipulates a variable T whose initial value is 0. Each time a process p_i sends a messages m, T is incremented by one, i.e. $T := T + 1$. The message m carries the value of T as $m.T$. On receipt of a message m from another process p_j, the variable T in the process p_i is manipulated as $T := max(T, m.T)$. In physical clock and liner clock $m_1.T < m_2.T$, if a message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$). However, "$m_1 \rightarrow m_2$" may not hold even if $m_1.T < m_2.T$. The message length is $O(1)$ independently of the size of the group.

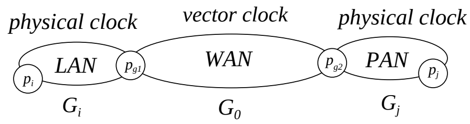

Figure 1. Structured group.

Suppose a group includes n processes $p_1, ..., p_n$. In a *vector clock* [6], each process p_i manipulates a vector $\langle T_1, ..., T_n \rangle$, where a value of each element T_j is initial 0 ($j = 1, ..., n$). Each time a process p_i sends a message m, the ith element T_i is incremented by one, i.e. $T_i := T_i + 1$ (i=1,...,n). Then, the message m carries the vector T of the sender process p_i as $m.T$(=$\langle m.T_1, ..., m.T_n \rangle$). On receipt of a message m from a process p_k, a process p_i manipulates the vector T as $T_k := max(T_k, m.T_k)$ ($k = 1, ..., n$, $k \neq j$). Here, a message m_1 *causally precedes* another message m_2 ($m_1 \rightarrow m_2$) if and only if (iff) $m_1.T < m_2.T$. m_1 is *causally concurrent* with m_2 ($m_1 \parallel m_2$) iff $m_1.T$ and $m_2.T$ are not comparable, i.e. neither $m_1.T \leq m_2.T$ nor $m_1.T \geq m_2.T$. Since a message length is $O(n)$ in the vector clock, it is not easy to use the vector clock for a large group which includes a large number of processes.

In this paper, a group G is composed of subgroups $G_1, ..., G_k$. Each subgroup G_i adopts some type of clock *clock*(G_i) $\in \{$ *RT(real time)*, *LT(liner clock)*, *VT(vector clock)* $\}$ (i=1,...,k). A pair of different subgroups G_i and G_j may use different types of clocks. For example, real time supported by physical clock is used to causally order messages in a network like PAN and LAN since the delay time is shorter as shown in Figure 1. A logical clock like liner clock and vector clock is used in a network like WAN where the delay time is longer. Subgroups with physical, liner, and vector clocks are referred to as RT, LT, and VT subgroups, respectively.

Suppose there are a pair of subgroups G_i and G_j in each of which processes are interconnected in a local area network where the maximum delay time is about one [msec]. Gateway processes of G_i and G_j are interconnected with a wide-area network G_0. In this paper, we consider a group where processes in the subgroups G_i and G_j take usage of physical clock and linear clock, respectively, i.e. RT and LT subgroups. The vector clock is used to exchange messages among

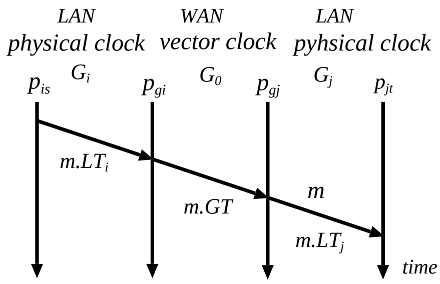

Figure 2. Time-stamp of messages.

gateway processes. Each message m is assigned with time-stamp $m.T$ showing "time" when m is sent. Processes in the subgroups G_i and G_j deliver messages in time-stamp order.

We discuss how to causally deliver messages to processes in subgroups by using vector clock, liner clock, and physical clock. Let p_{gi} and p_{gj} be a pair of gateway processes of subgroups G_i and G_j, respectively. In a group of gateway processes p_{gi} and p_{gj} communicate with one another in a wide-area network (WAN). Each gateway process delivers messages by using the vector clock.

Suppose a process p_{is} in a subgroup G_i sends a message *m* to another process p_{jt} in a subgroup G_j $\left[\text{Figure 2}\right]$. First, a gateway process p_{gi} receives the message *m* with time-stamp $m.LT_i$ which is a physical clock. The gateway p_{gi} forwards the message *m* with time-stamp $m.GT$ which is a vector clock, to another gateway p_{gj}. The gateway process p_{gj} receives the message *m* with $m.GT$ and then forwards *m* to the process p_{jt} in G_j. Here, the message has local time-stamp $m.LT_j$ which is a physical clock.

3. Clock Synchronization

3.1 Local network

In a local network whose delay time between every pair of processes is shorter than one [msec], processes deliver messages by using physical clock. Here, processes have to synchronize physical clocks with each other.

3.2 Global network

In a global network like the Internet, the delay time is about 100 times longer than a local network. In these networks, it is difficult to synchronize physical clocks of processes at high accuracy. Local subgroups are interconnected in a global network. Each process in a local subgroup uses physical clock. A process in each subgroup named a $gateway$ process communicates with $gateway$ processes of the other subgroups in the global network.

Processes in a global subgroup use the vector clock to deliver messages. A global subgroup is a *VT* type and a local subgroup is an *RT* type. A gateway process translates vector clock in the global subgroup and real time in a local subgroup.

A gateway process assigns a message with time-stamp of vector clock when sending the message in a *VT* subgroup. The gateway process assigns a message time-stamp of physical clock when sending the message in an *RT* subgroup. Messages sent by a process in the *RT* subgroup to a gateway process are totally ordered by physical clock.

4. Causality on Heterogeneous Clocks

4.1 Structured group

We discuss what kinds of difficulties to occur to causally deliver messages through subgroups with heterogeneous clocks in a group. Suppose a group G is composed of a global subgroup G_0 and multiple local subgroups G_1, ..., G_k ($k \geq 2$). Each local subgroup G_i is composed of a *gateway* process p_{i0} and normal processes p_{i1}, ..., p_{il_i} ($l_i \geq 1$)($i = 1, ..., k$). A gateway process p_{i0} communicates with gateway processes of the other subgroups. A global subgroup G_0 is composed of gateway processes. We make following assumptions:

1. Processes in each local subgroup G_i are interconnected with a local network like local area network and personal area network ($i = 1, ..., k$).
2. Gateway processes in a global subgroup G_0 are interconnected with a wide-area network like the Internet.

In each local subgroup G_i, every process uses the same type of clock, physical clock or liner clock. If the processes are interconnected with a personal area network and one-hop ad hoc network [3], the processes use the physical clock to synchronize message communication since delay time between every pair of processes is so short that variance among clocks in different processes can be neglected.

In a real time (*RT*) subgroup G_i, a process p_{ij} assigns a message *m* with time-stamp, i.e. current time $m.T$ shown by the physical clock on sending the message *m*. This means all the messages transmitted in the subgroup G_i are totally ordered in the time-stamps.

On the other hand, the vector clock is used in the global subgroup of gateways which are interconnected in the Internet. Messages transmitted in the global subgroup are referred to as *global* while messages transmitted in local subgroups are *local*. *global* messages are partially ordered, i.e. causally ordered in the vector clock while local messages are totally ordered in each local subgroup.

Suppose a pair of messages m_1 and m_2 are transmitted in a local subgroup G_i. Here, m_1 is defined to *locally causally precede* m_2 in G_i ($m_1 \rightarrow_i m_2$) iff a sending event of m_1 happens before m_2 in G_i. Next, suppose a pair of messages m_1 and m_2 are transmitted in different local subgroups G_i and G_j, respectively. m_1 *globally causally precedes* m_2 ($m_1 \rightarrow m_2$) iff a sending event of m_1 happens before m_2.

4.2 Causality

First, suppose a process p_{is} in a local subgroup G_i sends a local message *m* with time-stamp $m.LT$. It is noted that a pair of local messages m_1 and m_2 in a local subgroup G_i may be causally concurrent even if $m_1.LT < m_2.LT$ or $m_1.LT > m_2.LT$ as presented in the preceding subsection. On receipt of the message *m*, the gateway process p_{i0} in G_i forwards *m* to other gateway processes in the global subgroup G_0. Here, the message *m* is assigned with the vector clock which is shown by $m.GT$. Next, a gateway process p_{j0} in a local subgroup G_j receives the message m. The gateway process p_{j0} forwards the message m to local processes in G_j. Here, the message m is time-stamped with $m.LT$ showing local time when the gateway process p_{j0} sends m in the local subgroup G_j. *Local time* means real time or linear time. Finally a process p_{jt} in a local subgroup G_j receives a message m_1 which is sent by a process p_{is} in the local subgroup G_i as shown in Figure 3.

Suppose a process p_{ju} in the local subgroup G_j sends a message m_2 where $m_2.LT < m_1.LT$. $m_1.LT$ shows local time when the gateway process p_{j0} sends a local message m_1 in the local subgroup G_j. Here, m_2 *locally causally precedes* m_1 ($m_2 \rightarrow_j m_1$) in G_j. However, the source process p_{is} does not send the message m_1 after receiving the message m_2. Hence, m_1 and m_2 are causally concurrent in a group G ($m_1 \| m_2$) even if $m_2 \rightarrow m_1$ in the subgroup G_j.

A gateway process p_{j0} receives a message m from another gateway process p_{i0}. Then, the gateway process p_{j0} forwards the message m with time-stamp $m.LT_j$ in a local subgroup G_j. Suppose $m.LT_j$ shows real time in the local subgroup G_j. As shown in Figure 3, the gateway process p_{j0} cannot assign the message m with the current time as its sending time because m had been already sent in the local subgroup G_i. Let δ_i be the minimum delay time in a local subgroup G_i and δ be the minimum delay time in a global subgroup G_0. The gateway process p_{j0} assigns the message m with the time-stamp $m.LT_j$, $m.LT_j := T - \delta - \delta_j$. Here, *T* shows current time of the gateway process p_{j0}. It is straightforward for the following theorem to hold from the definitions:

[Theorem] Let m_1 and m_2 be messages in a local subgroup G_i. A message m_1 globally causally precedes another message m_2 ($m_1 \rightarrow m_2$) only if $m_1.LT_i < m_2.LT_i$. □

As pointed out here, a pair of local messages m_1 and m_2 are totally ordered by using the physical clock even if m_1 and m_2 are causally concurrent in a local subgroup G_i. If a pair of messages m_1 and m_2 are sent

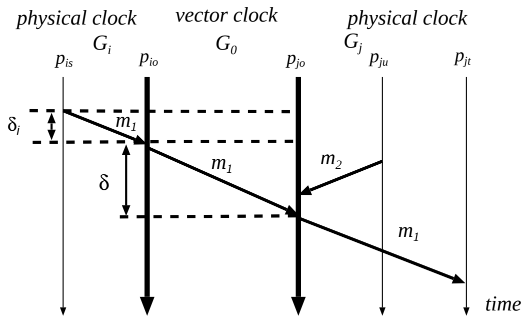

Figure 3. Causal precedency.

out to other local subgroups in a global subgroup G_0 of gateway processes, a pair of global messages m_1 and m_2 are ordered as well, i.e. $m_1.GT < m_2.GT$ if $m_1.LT < m_2.LT$. A vector clock GT is in a form $\langle GT_1, ..., GT_k \rangle$ where each element GT_i shows logical time of a gateway process p_{i0} of a local subgroup G_i (*i=1, ..., k*). Each time a gateway process p_{i0} sends a global message, the *i*th element GT_i in the vector *GT* is incremented by one independently of which local process sends the message in the local subgroup G_i. Hence, $m_1.GT < m_2.GT$ if and only if (iff) $m_1.LT < m_2.LT$. In the local subgroup G_i, every pair of local processes p_{is} and p_{it} are equipped with physical clock c_{is} and c_{it}, respectively, showing the *same* time, i.e $|c_{is}(t) - c_{it}(t)| \leq \delta_c$ where δ_c is the maximum allowable difference in the two physical clocks c_{is} and c_{it} in the local subgroup. $c_{it}(t)$ denote time value of clock c_{it} where *t* is UTC time. Hence, a following property holds:

[Property] A pair of local messages m_1 and m_2 are causally concurrent ($m_1 \| m_2$) if $|m_1.LT - m_2.LT| < \delta_c$. □

Each gateway process p_{i0} in a local subgroup G_i has a local message log LML_i which holds messages which p_{i0} has received from local processes in G_i are stored.

The vector clock $GT = \langle GT_i, ..., GT_k \rangle$ is manipulated in a gateway process p_{i0} of a local subgroup G_i (*i=1,...,k*) as follows:

- On receipt of a local message *m*, $m.GT_i := m'.GT_i$ +1 where m' is a message whose time-stamp is the smallest in the local message log LML_i and where $|m.LT - m'.LT| \geq \delta_c$.

[Theorem] Let m_1 and m_2 be messages sent in a local subgroup G_i. m_1 is causally concurrent with m_2 ($m_1 \| m_2$) if $m_1.GT_i = m_2.GT_i$. □

[Theorem] Let m_1 and m_2 be messages sent in a local subgroup G_i. m_1 causally precedes m_2 ($m_1 \rightarrow m_2$) only if $m_1.GT_i < m_2.GT_i$. □

A process p_{it} sends a message m_1 and another process p_{is} sends a message m_2 after receiving m_1 in a local subgroup G_i. A gateway process p_{i0} receives m_2

after it takes a large time than δ_i time units since p_{i0} receives m_1. Here, $m_2.GT_i := m_1.GT_i + 1$. If p_{i0} receives m_2 in δ_i time units, $m_2.GT_i := m_1.GT_i$.

In Figure 4 and Figure 5, a gateway process p_{j0} receives a message m_1. After sending a message m_1 to a process in a local subgroup G_j, the gateway process p_{j0} receives a message m_2. Here, $m_2.GT_j := m_1.GT_j + 1$. The gateway process p_{j0} receives a global message m_1 and forwards m_1 to processes in the local subgroup G_j. Then, the gateway process p_{j0} receives a local message m_2 from a local process p_{js}. If $m_2.LT - m_1.LT > \delta_j$, p_{js} might send m_2 after receiving m_1, i.e. m_1 might causally precede m_2 as shown in Figure 4. Hence, $m_2.GT_j > m_1.GT_j$. On the other hands, if $m_2.LT - m_1.LT \leq \delta_j$, it is sure p_{js} sends m_2 before receiving m_1. That is, m_1 and m_2 are causally concurrent as shown in Figure 5. Here, $m_2.GT_j := m_1.GT_j$.

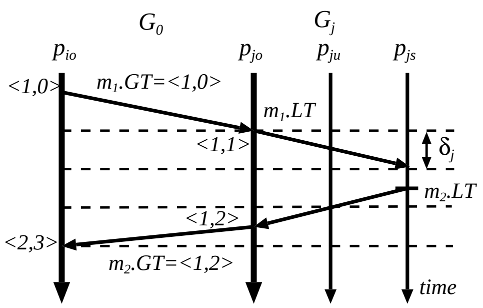

Figure 4. Causality of physical clock.

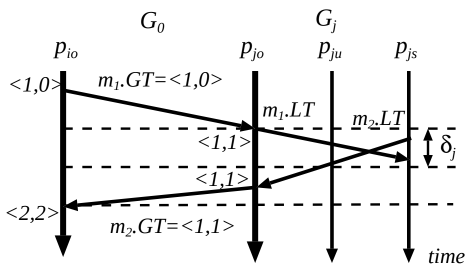

Figure 5. Causality of physical clock.

[Theorem] Let m_1 and m_2 be messages received and sent by a same process, respectively, in a local subgroup G_j. A message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$) only if $m_2.LT - m_1.LT > \delta_j$. □

[Theorem] Let m_1 and m_2 be messages received and sent by different processes, respectively, in a local subgroup G_j. A message m_1 causally precedes another message m_2 ($m_1 \rightarrow m_2$) only if $m_2.LT - m_1.LT > 2\delta_j$. □

5. Evaluation

We evaluate our protocol named HCG (heterogeneous clock group) protocol compared with the basic (B) protocol. The B protocol is the same as HCG protocol except that every message m_1 causally precedes another message m_2 if m_2 is sent after receiving m_1 in a gateway.

We measure how many messages are ordered in the HCG and B protocols. A local process p_{js} first sends messages in a local subgroup G_j. A gateway process p_{gj} receives the messages and then forwards the messages to other gateway processes in a global subgroup. Here, we assume that each of gateway process p_{gj} and a process p_{js} in a local subgroup sends randomly one message every λ time units. Suppose a gateway process p_{gj} sends a message a and a local process sends a message b in a local subgroup G_j. On receipt of a message b from the local process p_{js}, the gateway process p_{gj} compares a time-stamp $b.LT$ with a time-stamp $a.LT$. If $|b.LT - a.LT| > \delta_j$, the gateway process p_{gj} considers that the message a causally precedes the message b ($a \rightarrow b$) only if $|b.LT - a.LT| > \delta_j$.

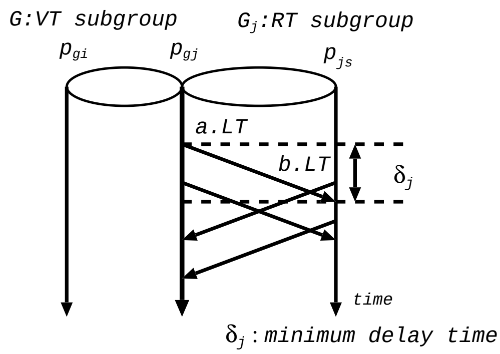

Figure 6. Evaluation model.

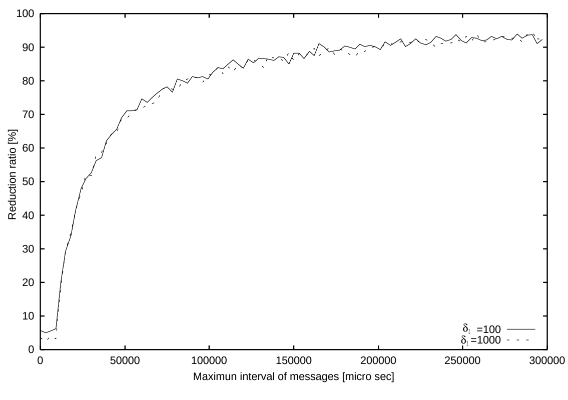

Figure 7. Reduction ratio of causally order messages.

In Figure 7, the vertical axis shows the *reduction* ratio[%] of the messages causally ordered in the

HCG protocol to the B protocol for λ (100 to 300000 μseconds). For example, "$\lambda = 100$" means that each process sends one message every λ time units [μsec] where $0 \leq \lambda \leq 100$. Figure 7 shows the message reduction ratio for $\delta_l = 100$, 1000 [μsec]. In a local area network, the delay time between every pair of processes is shorter than one [msec]. "δ_l = 100 [μsec]" shows the delay time of a personal or local area network. For λ = 30000 and $\delta_l = 1000$, only about 50% of messages which a local process sends causally precede messages which a gateway process sends. The more frequently a process sends messages, the fewer number of messages are causally ordered. Figure 8 shows the message ratio vs. the delay time δ_l for λ= 30000. "λ= 30000" means that a getaway process sends a message to local processes every λ time units [μsec] and receives a message from local processes every λ time units [μsec] ($0 \leq \lambda \leq 30000$). Figure 8 shows that the reduction ratio of the HCG protocol to the B protocol is invariant for the delay time δ_l, about 45%.

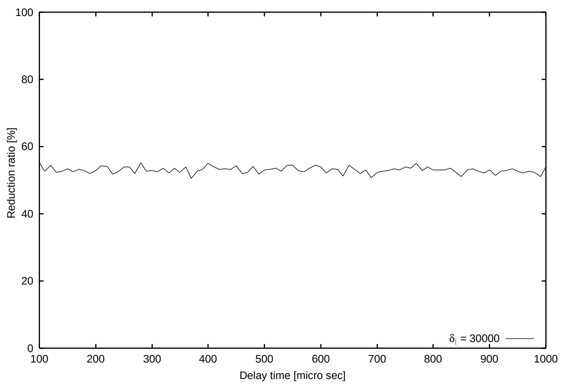

Figure 8. Reduction ratio and delay time.

6. Concluding Remarks

In distributed applications like P2P, a large number possibly millions of processes distributed in a wide area are required to be cooperating. The vector clock can not be used for a large-scale group due to the message length and computation overhead. In this paper, we proposed a structured group where local subgroups are interconnected with the Internet and local processes are interconnected in local and personal area networks. In addition, the vector clock is only used in the Internet. Local processes in each local subgroup use physical clock or liner clock since the delay time is so short that clocks in every computer can be synchronized with the others. We discussed how to causally order messages exchanged among subgroups with different clocks, i.e. vector, and liner, and physical clock. If messages are ordered according to the synchronization mechanism in each group, some message m_1 is ordered to precede another message m_2 even if m_1 and m_2 are causally concurrent. In this paper, we discussed how prevent from unnecessary ordering of messages. We showed the number of messages to be ordered can be decreased in our protocol named HCG protocol than the traditional vector clock protocol.

References

[1] K. P. Birman and R. V. Renesse. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1993.

[2] Bluetooth SIG, Inc. *Bluetooth V1.1 Core Specifications*, 2001.

[3] L. Huang, T. Lai, and D. Zhou. On the Scalability of IEEE 802.11 Ad Hoc Networks. *Proc. of The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002)*, pages 172–181, 2002.

[4] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Comm. ACM*, 21(7):558–565, 1978.

[5] LAN MAN Standards Committee of the IEEE Computer Society. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher speed Physical Layer (PHY) Extension in the 2.4 GHz Band.*, Sept. 2001.

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

[7] D. L. Mills. Network Time Protocol. RFC 1350, 1992.

[8] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal. Extended Virtual Synchrony. *Proc. of IEEE ICDCS-14*, pages 56–65, 1994.

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

[10] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly & Associates*, 2001.

[11] T. Peatfield. *Network Systems Tutorial for IEEE Std 802.3: Repeater Functions and System Design Topology Considerations for Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Local Area Networks (LANs)*. Inst of Elect & Electronic, 1995.

[12] R. Prasad. Basic Concept of Personal Area Networks. *WWRF, Kick off Meeting*, 2001.

[13] J. Pstel. Internet Protocol. RFC 791, 1981.

[14] J. Psterl. Transmission Control Protocol. RFC 793, 1981.

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