Multimedia Communication in a Hierarchical Group

Yasutaka Nishimura, Tomoya Enokido, and Makoto Takizawa
Dept. of Computers and Systems Engineering
Tokyo Denki University
E-mail {yasu, eno, taki}@takilab.k.dendai.ac.jp

Large number of peer processes are cooperating by exchanging messages in peer-to-peer systems. In this paper, we discuss a hierarchical group protocol aiming at reducing communication and computation overheads for scalable group communication. A hierarchical group is composed of subgroups where each subgroup is furthermore composed of subgroups. In traditional hierarchical groups, a pair of subgroups communicate with one another through a gateway process. A gateway process is performance bottleneck and single point of failure. In order to increase the throughput and reliability of inter-subgroup communication, messages are in parallel transmitted in a striping way through multiple channels between the subgroups. We discuss how to design a hierarchical group for realizing high-performance multimedia communication among large number of peer processes.

階層型グループにおけるマルチメディアグループ通信

西村 康孝　榎戸 智也　滝沢 誠
東京電機大学大学院理工学研究科情報システム工学専攻
E-mail {yasu, eno, taki}@takilab.k.dendai.ac.jp

Peer-to-peer システムに代表される大規模分散システムは多数のプロセスが相互接続し、メッセージを送受信することにより協調動作を行う。大規模分散マルチメディア環境において多対多の通信を行う場合、ネットワークや各プロセスにかかる負荷が問題となる。そこで、多数のプロセスをいくつかのグループに分割し、多階層のグループから大規模な通信を行うことで通信の負荷を軽減する手法がある。本論文では、階層型グループにおいてボトルネックとなるゲートウェイプロセスを必要としない新しい方式を提案し、その構成法を論じる。また、評価を行う。

1. Introduction

In various types of applications, multimedia messages are exchanged among application processes. Each application requires a system to support some quality of service (QoS), bandwidth, delay time, and message loss ratio. It is critical to discuss how to support each of huge number and types of application processes with enough QoS in change of network environments and requirements. In this paper, we discuss how to support flexible group communication service of multimedia data for applications. In peer-to-peer (P2P) [16] and Grid [9] computing systems, hundreds to thousands, possibly million peer processes are cooperating, which are widely distributed in networks. In a *wide-area* group, processes are distributed in wide-area networks. Takizawa *et al.* [22] discuss fully distributed protocols for a wide-area group which supports destination retransmission to reduce time for detecting and retransmitting messages lost.

Traditional communication protocols, TCP [18] and RTP [19] support processes with reliable one-to-one and one-to-many transmission of data, respectively. Here, messages are efficiently and reliably transmitted from a process to one or more than one destination process. Recently, multiple connections are used to in parallel transmit data from a process to another process in the *network striping* like GridFTP [2], SplitStream [6], and PSockets [20] in order to increase the throughput. In PSockets [20], data is divided into partitions and the data is striped over multiple sockets, i.e. each partition is transmitted at a different socket. In SplitStream [6], data is split and each of split data is transmitted in a tree routing. In GridFTP [2], a high-performance file transfer protocol (FTP) is discussed by using multiple connections.

Tree routing protocols [7, 10] to multicast messages are discussed. In the group communication, processes not only send messages to but also receive messages from multiple processes. Various types of group communication protocols are discussed to causally deliver messages [15]. Takizawa and Takamura [24] discuss how to support the causally ordered delivery of messages in a hierarchical group by using the vector clock whose size is the total number of processes. Here, a group is composed of subgroups where processes in different subgroups exchange messages via gateway processes. Taguchi and Takizawa [23] discuss two-layered and multi-layered group protocols where a group is composed of subgroups. In Totem [14], processes are interconnected in a ring network. Rings can be hierarchically interconnected. Here, messages are ordered by using the token passing mechanism. The protocol cannot be adopted for a large-scale group due to delay time to pass a token.

In these hierarchical protocols, a gateway process in one subgroup exchanges messages with other subgroups. Each gateway process is not only performance bottleneck but also single point of failure. In this paper, we discuss a *hierarchical group* (*HG*) for a large-scale, wide-area group of processes for supporting high-performance multimedia communication. Here, a pair of subgroups are interconnected through multiple communication channels among multiple processes in the subgroups to realize parallel, striping communication [20] and to increase the reliability. That is, inter-subgroup communication is a many-to-many type. In addition, the number of connections can be changed, i.e. the more number of connections are used, the higher bandwidth and reliability are supported for applications. In order to transmit multimedia data, realtime constraints are satisfied. We discuss how to design a hierarchical group for a set of processes so as to realize the delay time among processes.

In section 2, we discuss a model of a hierarchical group (HG). In section 3, we discuss striping inter-group communication. In section 4, we discuss how to design hierarchical group. In section 5, we evaluate the hierarchical group in terms of delay time and the number of messages compared with the flat group.

2. Hierarchical Group

2.1. Model of hierarchical group

A *group* of multiple peer processes are cooperating by exchanging messages in order to achieve some objectives. In one-to-one and multicast communications [7], each message is *reliably* routed to one or more than one process in tree routing protocols [18]. On the other hand, a process sends a message to multiple processes while receiving messages from multiple processes in group communication [4,5]. Here, 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* [12] a sending event of m_2 [4]. Here, every common destination process of messages m_1 and m_2 is required to deliver the message m_1 before the other message m_2. Linear clock [12] and vector clock [13] are used to causally deliver messages in distributed systems.

In a *flat* group, every pair of peer processes directly exchange messages with one another. Most group communication protocols [5,15,21] are discussed for flat groups. Due to computation and communication overheads $O(n)$ to $O(n^2)$ for the number n of processes in a flat group, it is difficult to support a large number of processes with group communication service. In order to realize a scalable group, we discuss a *hierarchical group* G which is composed of subgroups. Processes in a group G are partitioned into multiple subgroups. There is one *root* subgroup G_0. Subgroups $G_1, \ldots, G_k$ are connected to the root subgroup G_0. Here, k indicates the degree of the root subgroup G_0, i.e. the number of child subgroups. Then, each subgroup G_i is furthermore connected with subgroups $G_{i1} \ldots G_{ik_i} (i = 1, \ldots, k)$ as shown in Figure 1. A subgroup G_i is composed of processes $p_{i1}, \ldots p_{is_i} (s_i \geq 1)$ where s_i is the size of the subgroup G_i. Here, G_i is a parent subgroup of G_{ij} which is in turn a child group of G_i according to the tree conventions. Thus, the subgroups are hierarchically structured. In a hierarchical group [23], every pair of parent subgroup G_i and child subgroup G_{ij} communicate with one another through one channel between gateway processes g_i and g_{ij} as shown in Figure 2. Here, the gateway processes and communication channel between them imply performance bottleneck and single point of failure. In order to increase the performance and reliability, every pair of parent and child subgroups communicate through multiple communication channels as shown in Figure 1.

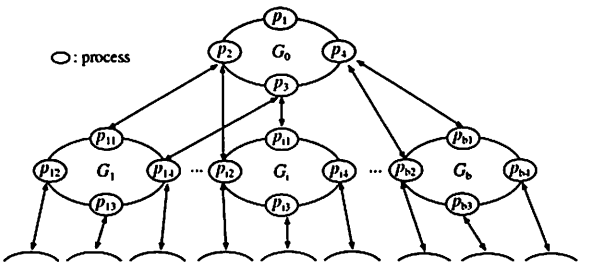

Figure 1. Hierarchical group.

For example, a process p_{i2} in a parent group G_i communicates with a pair of processes p_{ij1} and p_{ij2} in a child subgroup G_{ij}, and another process p_{i3} communicates with processes p_{ij1} and p_{ij4} in G_{ij} as shown in Figure 3. The processes p_{i2}, p_{i3}, p_{ij1}, p_{ij2}, and p_{ij4} play a role of gateway between a pair of the subgroups G_i and G_{ij}. Thus, a gateway process in the parent subgroup G_i is interconnected with gateway processes p_{ij1} and p_{ij2} in the child subgroup G_{ij} through multiple channels. A gateway process in a child subgroup G_{ij} has also multiple channels with gateway processes in a parent subgroup G_i. A pair of parent and child subgroups are interconnected with many-to-many communication among gateway processes. Thus, a subgroup G_{ij} communicates with one parent subgroup G_i and child subgroups $G_{ij1} \ldots G_{ijk_{ij}} (k_{ij} \geq 0)$. A process p_{ij} in a subgroup G_{ij} which communicates with processes in other subgroups are named *gateway* processes. Gateway processes communicating with the parent subgroup G_i and child subgroup G_{ijh} are referred to as *upward* and *downward* gateway processes, respectively. Each process can be both types of gateways. In Figure 3, processes p_{ij1} and p_{ij2} are upward gateway processes in a subgroup G_{ij}, and processes p_{i2}, p_{i3}, and p_{i4} are downward gateway processes in a subgroup G_i. Normal processes are ones which are not gateways. p_{i2} and p_{i4} are normal processes. Gateway processes also have functions for transmitting messages in a same way as normal processes. In a *root* subgroup, there are normal processes and only downward gateway processes. A *leaf* subgroup includes normal processes and only upward gateway processes. If all the leaf subgroups are at the same layer of the hierarchy, the hierarchical group is *height-balanced*.

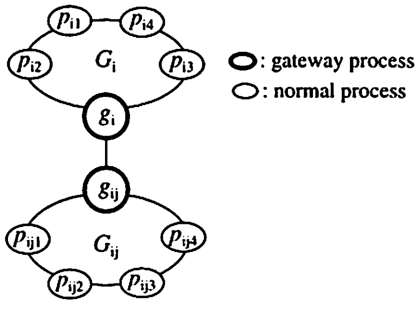

Figure 2. Inter-subgroup communication.

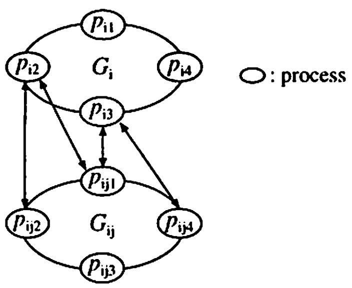

Figure 3. Striping communication.

Since the communication and computation overheads are $O(n^2)$ for number n of processes in a group, the size of each subgroup is bounded due to the limitted computation capacity of each process. The number s_i of processes in a subgroup G_i is bounded to be smaller than $S(s_i \leq s)$. The smaller size of each subgroup is, the more number of subgroups, i.e. the height or breadth is increased. If the number b_i of child subgroups of a subgroup G_i is increased, the overhead for inter-group communication is increased. Hence, $s_i \geq s$ when s shows the minimum number of processes in the subgroup G_i. Processes leave and join a subgroup G_i e.g. due to the fault and recovery from the fault. In addition, quality of service (QoS) supported by processes and networks is changed. Processes in a subgroup may move to another subgroup to satisfy the performance and QoS requirements. If the number of the processes is larger than S, the subgroup G_i is splitted. On the other hand, if the number of a subgroup G_i gets smaller than s, the subgroup G is merged into a sibling subgroup. Thus, $S \geq s_i \geq s$ for the size s_i of every subgroup G_i. A hierarchical group is dynamically height-balanced as discussed in B-tree [3].

2.2. Data transmission

In this paper, we assume a process sends a message to all the processes, i.e. broadcasts a message in a group G. Suppose a process p_{ij} originally transmits a message m in a subgroup G_{ij}. The messages are forwarded to processes in the group G as follows :

1. The process p_{ij} first sends a message m to every process in the subgroup G_{ij}.
2. On receipt of a message m, an upward gateway process p_{ih} forwards the message m up to downward gateway processes in the parent subgroup G_i.
3. On receipt of a message m, a downward gateway process p_{ih} forwards the message m down to upward gateway processes in child subgroups $G_{ij1}, \ldots G_{ijk_{ij}}$.

In each subgroup, a process delivers messages to all the processes by using its own synchronization mechanism like vector clock [13]. Gateway processes in parent and child subgroups communicate with each other in the striping transmission way [6]. The striping inter-subgroup communication is discussed later.

3. Striping Inter-subgroup Communication

In traditional hierarchical groups [8, 23], processes in a pair of subgroups G_i and G_{ij} communicate with each other only through one gateway process in each subgroup as shown in Figure 2. Here, the gateway process and channel between the gateway processes can be performance bottleneck. In addition, if the gateway process or the channel is faulty, the subgroups G_i and G_{ij} cannot be communicated, i.e. single point of failure. In order to increase the performance and reliability of group communication, a pair of parent and child subgroups communicate with one another through multiple channels. Let D_{ij} be a set of downward gateway processes in a parent subgroup G_i to communicate with a child subgroup G_{ij}. Let U_{ij} be a set of upward gateway processes in a child subgroup G_{ij} to communicate with a parent subgroup G_i. The downward gateway processes in D_{ij} are communicating upward with the processes in U_{ij} in a many-to-many type of communication.

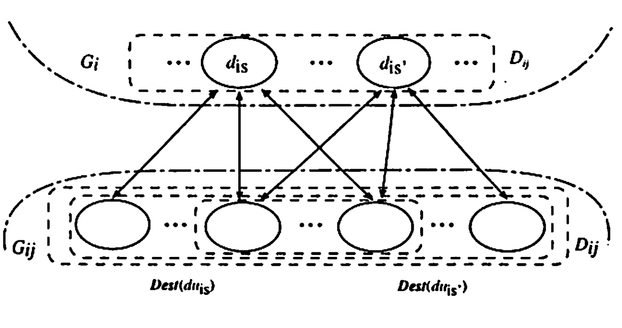

Figure 4. Striping.

Messages are transmitted in subgroups. Suppose a downward gateway process d_{is} in D_{ij} receives a message in a parent subgroup G_i. Each downward gateway processes d_{is} is connected with upward gateway processes in U_{ij}. Here, let $\textbf{\textit{DestU}}(d_{is})$ be a set of upward gateway processes in a child subgroup G_{ij} with which a downward gateway process d_{is} communicates. $\textbf{\textit{DestU}}(d_{is}) \subseteq D_{ij}$. There are following ways for downward gateway processes in a parent subgroup G_i to forward messages to the child subgroup G_{ij} :

1. Each gateway process sends same messages to the destination processes.
2. Each gateway process sends messages different from the other gateway processes.

In addition, each downward gateway process transmits messages to multiple upward gateway processes in $\textbf{\textit{DestU}}(d_{is})$. There are following ways for a downward gateway process d_{is} to transmit messages :

1. Same messages are transmitted to each upward gateway process in $\textbf{\textit{DestU}}(d_{is})$.
2. Different messages are transmitted to each process in $\textbf{\textit{DestU}}(d_{is})$.

In addition, each downward gateway in G_i can in parallel send messages to multiple upward processes in G_{ij}. That is, a gateway process sends different messages to different upward gateway processes.

Each upward gateway process in a child subgroup G_{ij} sends messages to downward gateway processes in a subgroup G_j in a same way as the upward-to-downward many-to-many communication.

4. Design of Hierarchical Group

We discuss how to design a hierarchical group for a set $\mathbf{G}$ of peer processes $p_1, \ldots, p_n$ which are distributed in networks. Here, the size $|\mathbf{G}|$ of the group $\mathbf{G}$ is n. Each pair of processes p_i and p_j can communicate with one another through a logical channel C_{ij}. A channel can be realized in UDP [17] or a connection of TCP [18]. Each channel C_{ij} is characterized in quality of service (QoS) Q_{ij}, i.e. delay time, bandwidth, and packet loss ratio. In this paper, we assume that each channel supports enough bandwidth like 10G Ethernet [1]. Messages may be lost and delayed due to congestions and faults. In order to realize real-time multimedia communications, it is critical to decrease the delay time. We discuss how to construct a hierarchical group from a set $\mathbf{G}$ of processes so as to minimize the delay time.

Let d_{ij} stand for the message delay time from a process p_i to another process p_j. The delay time d_{ij} can be obtained in networks, for example, by using the ping mechanism. The *distance* $\delta(p_i, p_j)$ between a pair of processes p_i and p_j is defined to be round trip time $d_{ij} + d_{ji}$ between p_i and p_j. $\delta(p_i, p_i) = 0$ for every process p_i. The distance is symmetric from the destination. Let $\mathbf{D_G}$ be a set of distances between every pair of processes in $\mathbf{G}$, $\{\delta(p_i, p_j) |\ p_i, p_j \in \mathbf{G}\}$. $\textbf{\textit{AvDist}}(p_i, \mathbf{G})$ shows the average distance from a process p_i to every other process in $\mathbf{G}$, i.e. $\sum_{p_j \in \mathbf{G}} \delta(p_i, p_j)/(|\mathbf{G}| - 1)$.

Given a process set $\mathbf{G}$ and the delay set $\mathbf{D_G}$, a parent subgroup $\mathbf{G_0}$ and child subgroups $\mathbf{G_1}, \ldots, \mathbf{G_k}$ are obtained by the following procedure $\textbf{\textit{DV}}$ where s is the number of processes to be in G_0 and k is the number of child subgroups of G_0. Here, $\mathbf{G} = \mathbf{G_0} \cup \mathbf{G_1} \cup \cdots \cup \mathbf{G_b}, \mathbf{G} \cap \mathbf{G_i} = \phi$, and $\mathbf{G_i} \cap \mathbf{G_j} = \phi$ for every pair of different subgroups $\mathbf{G_i}$ and $\mathbf{G_j}$.

$\textbf{\textit{DV}}(\mathbf{G}, \mathbf{D_G}, s, k)$ {
  $\mathbf{G_0} := \textbf{\textit{Parent}}(\mathbf{G}, \mathbf{D_G}, s)$;
  $\{\mathbf{G_1}, \ldots, \mathbf{G_k}\} := \textbf{\textit{Child}}(\mathbf{G} - \mathbf{G_0}, \mathbf{D_{G-G_0}}, k)$;
  if $\mathbf{G} = \mathbf{G_0}$,
    for $i = 1, \ldots, k$, {
    $\mathbf{G_{i0}} = \textbf{\textit{DV}}(\mathbf{G_i}, \mathbf{D_G}, s)$;
    $\mathbf{G_{i0}}$ is a child of $\mathbf{G_0}$}
  return($\mathbf{G_0}$);
  }

First, a parent subgroup $\mathbf{G_0}$ is obtained by the procedure $\textbf{\textit{Parent}}(\mathbf{G}, \mathbf{D_G}, s)$, where $\mathbf{G_0}$ includes more number of processes than $s/2 - 1$ and fewer number of processes than $s + 1$. The procedure $\textbf{\textit{Parent}}$ is given as follows :

1. Initially, $\mathbf{G_0} := \phi$; and $i := 0$;
2. Select a process p in whose $\textbf{\textit{AvDist}}(p, \mathbf{G})$ is the minimum in $\mathbf{G}$.
3. If $i = s$, return $(\mathbf{G_0})$;
4. If $i < s/2$, $\{\mathbf{G_0} := \mathbf{G_0} \cup \{p\}$; $\mathbf{G} := \mathbf{G} - \{p\}$; $i := i + 1$; go to 2;}
5. If $\textbf{\textit{AvDist}}(p, G) < \textbf{\textit{AvDist}}(p', G_0) + \alpha$ where p' is a process whose $\textbf{\textit{AvDist}}(p', G_0)$ is the minimum in $\mathbf{G_0}$, $\{\mathbf{G_0} := \mathbf{G_0} \cup \{p\}$; $\mathbf{G} := \mathbf{G} - \{p\}$; $i := i + 1$; go to 2.}
6. Return $(\mathbf{G_0})$;

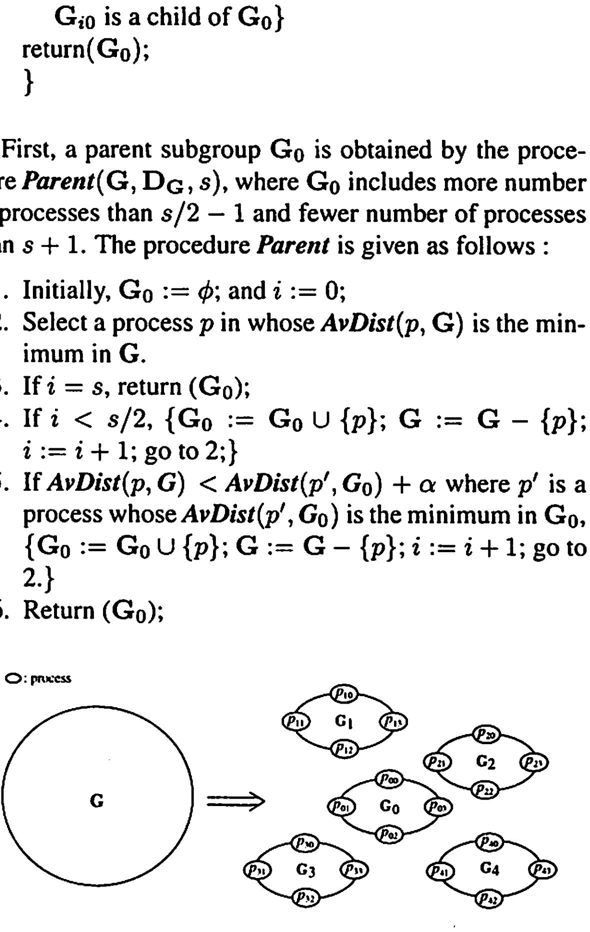

Figure 5. *k*-partitioning of group.

Here, α is a constraint. The larger α is, the more distant processes included in a subgroup. $\mathbf{G_0}$ is removed from $\mathbf{G}$. Then, the group $\mathbf{G}$ is partitioned into k subgroups $\mathbf{G_1}, \ldots, \mathbf{G_k}$ which to be child subgroups of $\mathbf{G_0}$ by the $\textbf{\textit{Child}}$ procedure using a type of k-medoids algorithm [11] to partition a group to k subgroups as shown in Figure 5. Processes which are nearer to each other in a group $\mathbf{G}$ are grouped into one subgroup. That is, $\delta(p_i, p_j) < \delta(p_i, p_k)$

for every pair of processes p_i and p_j in a subgroup $\mathbf{G}_i$ and every process p_k not in $\mathbf{G}_i$. There are algorithms like PAM (Partitioning Around Medoids) [11] and CLARA (Clustering LARge Applications) [11] to partition a collection of data into clusters. PAM is efficient for small number of processes ($n < 100$) and CLARA can be adopted for more number of processes. The algorithm PAM is briefly shown as follows :

Algorithm PAM

1. Select k representative processes arbitrarily.
2. Compute the *total cost* (TC_{ij}) for every pair of processes p_i and p_j where p_i is currently selected but p_j is not selected.
3. Select a non-selected process p_j whose total cost TC_{ij} is the minimum for the selected process p_i. If $TC_{ij} < 0$, replace p_i with p_j, i.e. p_j gets a selected process and p_j is not. Goto 2.
4. Otherwise, for each non-selected process p_j, find the most nearer representative process p_i and include p_j to a subgroup of p_i. Halt.

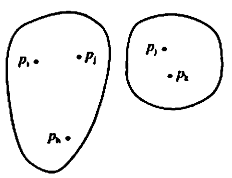

Figure 6. Four cases for replacing the medoid.

We discuss how to compute the total cost TC_{ij} for a pair of processes p_i and p_j. Let p_i be a current medoid which is to be replaced, p_h show the new medoid with which p_i is replaced, p_j denote another non-selected process which may or may not need to be changed in the subgroup, and p_k denote a current medoid which is nearest to p_j. The cost C_{ik} is first computed as follows [Figure 6] :

1. Suppose a process p_j in a subgroup of a selected process p_i and second similar selected process such that $\delta(p_j, p_h)$ is the minimum for every selected process.
   (a) $C_{jih} = \delta(p_j, p_k) - \delta(p_j, p_i)$ if $\delta(p_j, p_h) \geq \delta(p_j, p_k)$.
   (b) $C_{jih} = \delta(p_j, p_h) - \delta(p_j, p_i)$ if $\delta(p_j, p_h) < \delta(p_j, p_k)$.
2. Suppose a process p_j currently belongs to a subgroup other than the one represented by p_h. Let p_k be the representative process of that subgroup.
   (a) $C_{jih} = 0$ if $\delta(p_j, p_k) > \delta(p_j, p_h)$.
   (b) $C_{jih} = \delta(p_j, p_h) - \delta(p_j, p_k)$ if $\delta(p_j, p_h) < \delta(p_j, p_k)$.
   The total cost TC_{ih} is given $\sum_j C_{jih}$.

Next, the algorithm CLARA is shown as follows :

Algorithm CLARA

1. For $i := 1$ to 5, repeat the following steps:
2. Arbitrarily select a sample set S of $40 + 2k$ processes from a group **G**, and call the algorithm PAM to find k medoids of the sample set S.
3. For each process p_j in the group **G**, determine which of the k medoids is the most nearer to p_j and add p_j to the subgroup of the medoid.
4. Calculate the average distance of the subgroup obtained in the previous step. If this value is less than the current minimum, use this value as the current minimum, and retain the k medoids obtained so far.
5. Return to step 1 to start the next iteration.

The complexity of a single iteration is $O(k(n-k)^2)$ in PAM and $O(k(40+2k)^2 + k(n-k))$ in CLARA for n processes in a group **G**.

By using the procedure ***DV*** for a group **G** of processes, a hierarchical group $\mathbf{H_G}$ is obtained. The hierarchical group $\mathbf{H_G}$ is height-balanced.

5. Evaluation

We implement two versions $\boldsymbol{DV_p}$ and $\boldsymbol{DV_c}$ of the procedure ***DV*** which take usage of the PAM and CLARA algorithms to partition processes to subgroups, respectively. First, we measure how long it takes to obtain a hierarchical group $\mathbf{H_G}$ for a group **G** of n processes.

We measure the delivery time from a process to another process in a hierarchical group $\mathbf{H_G}$ and a flat group **G**. The delivery time is defined to be duration from time when a process starts transmitting a message until time when the message is delivered to all the destination processes.

In the simulation, n processes are randomly distributed to a geographical location in a 400×400 lattice. Here, one unit in the lattice shows a distance of one msecond delay time. The distance $\delta(p_i, p_j)$ between a pair of processes p_i and p_j is calculated in the Euclidean distance between the locations of p_i and p_j. The number s of processes in each subgroup is decided for the total number n of processes as follows :

1. if $n \leq 500$, $s = n/10$.
2. if $n \geq 500$, $s = 50$.

The height h of the group $\mathbf{H_G}$ is decided for n as follows :

1. $h = 10$ if $100 \leq n \leq 500$.
2. $h = \lfloor n/1000 \rfloor, n > 500$.

In the flat group **G**, a process directly sends a message $(n-1)$ times to deliver the message to $(n-1)$ processes. For example, it takes 52 mseconds to transmit 100 messges in a personal computer with dual Intel Pentium Xeon 1.8Ghz. If a process lastly sends a message to the most distant process, it takes the longest time. If a process lastly sends a message to the nearest process and every other process receives the message when the nearest process receives the message the delivery time is minimum. Figure 7 shows the maximum and minimum delivery time

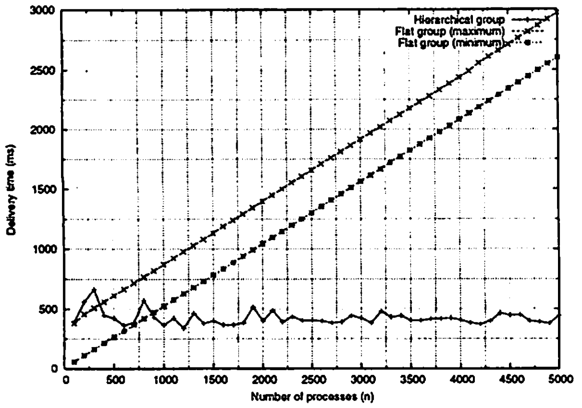

Figure 7. Delivery time.

in the flat group G and the longest delivery time in the hierarchical group $\mathbf{H_G}$. The delivery time of $\mathbf{H_G}$ is almost constant while the delivery time is $O(n)$ in the flat group.

6. Concluding Remarks

We discussed the hierarchical group (HG) where subgroups are hierarchically interconnected through gateway processes. In order to improve the reliability and throughput of the inter-subgroup communication, a pair of parent and child subgroup are interconnected through multiple communication channels between multiple gateway processes in the subgroup. We also discussed how to design a height-balanced hierarchical group fro ma set of processes. In the evaluation, we showed that the hierarchical group supports shorter delay time than the flat group.

References

[1] IEEE P802.3ae 10Gb/s Ethernet Task Force. *http://grouper.ieee.org/groups/802/3/ae/*, 2003.

[2] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data Management and Transfer in High-performance Computational Grid Environments. *Parallel Computing Journal*, 28(5):749–771, 2002.

[3] R. Bayer and E. M. McCreight. Organization and Maintenance of Large Ordered Indices. *Acta Informatica*, 1:173–189, 1972.

[4] K. Birman. Lightweight Causal and Atomic Group Multicast. *ACM Trans. on Computer Systems*, pages 272–290, 1991.

[5] K. P. Birman and R. v. Renesse. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1994.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Multicast in a Cooperative Environment. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP2003)*, pages 298–313, 2003.

[7] S. Deering. Host Groups: A Multicast Extension to the Internet Protocol. *RFC 966*, 1985.

[8] D. Dolev and D. Malki. The Transis Approach to High Availability Cluster Communication. *Communications of the ACM*, 39(4):64–70, 1996.

[9] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[10] M. Hofmann, T. Braun, and G. Carle. Multicast Communication in Large Scale Networks. In *Proc. of IEEE HPCS-3*, 1995.

[11] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

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

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

[14] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos. Totem: A Fault-Tolerant Multicast Group Communication System. *Communications of the ACM*, 39(4):54–63, 1996.

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

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

[17] J. Postel. User Datagram Protocol. *RFC 768*, 1980.

[18] J. Postel. Transmission Control Protocol. *Request for Comments 0793*, 1992.

[19] H. Schulzrinne, R. Casner, S. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-time Applications. *Request for Comments 1889*, 1996.

[20] H. Sivakumar, S. Bailey, and R. L. Grossman. PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks. In *Proc. of the 2000 ACM/IEEE Conference on Supercomputing*, page. http://www.sc2000.org/proceedings/techpapr/ papers/pap.pap240.pdf, 2000.

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

[22] T. Tachikawa, H. Higaki, and M. Takizawa. Δ-Causality and ε-Delivery for Wide-Area Group Communications. *Computer Communications Journal*, 23(1):13–21, 2000.

[23] K. Taguchi, T. Enokido, and M. Takizawa. Causal Ordered Delivery for a Hierarchical Group. In *Proc. of of IEEE 10th International Conference on Parallel and Distributed Systems (ICPADS2004)*, pages 485–492, 2004.

[24] M. Takizawa, M. Takamura, and A. Nakamura. Group Communication Protocol for Large Group. In *Proc. of IEEE LCN-18*, pages 310–319, 1993.