Journal of Mobile Multimedia, Vol. 1, No. 3 (2005) 180–197

DESIGN OF A HIERARCHICAL GROUP TO REALIZE A SCALABLE GROUP

YASUTAKA NISHIMURA, NAOHIRO HAYASHIBARA
Dept. of Computers and Systems Engineering, Tokyo Denki University
Saitama 350-0394, Japan
{yasu, haya}@takilab.k.dendai.ac.jp

TOMOYA ENOKIDO
Faculty of Business Administration, Rissho University
Shinagawa-ku, Tokyo 141-8602, Japan
eno@ris.ac.jp

MAKOTO TAKIZAWA
Dept. of Computers and Systems Engineering, Tokyo Denki University
Saitama 350-0394, Japan
taki@takilab.k.dendai.ac.jp

Received May 12, 2005
Revised Aug 12, 2005

According to the advance of computer and network technologies, information systems are getting scalable. Especially, peer-to-peer (P2P) overlay networks and Grid computing system are now taking a central position in information systems. In these systems, a large number of peer processes are cooperating. In group communication, each peer process sends a message to multiple processes while receiving messages from multiple processes. Here, messages transmitted are required to be causally/totally delivered to every common destination of the messages. The computation and communication complexity is $O(n)$ to $O(n^2)$ for the number n of peer processes. In order to reduce the overheads, a group is divided into smaller subgroups where processes exchange messages with other subgroups only through gateway processes while processes directly exchange messages in each subgroup. In this paper, we discuss a hierarchical group protocol aiming at reducing communication and computation overheads for supporting a scalable group of cooperating peer processes. In traditional hierarchical group protocols, each subgroup communicates with another subgroup through a single gateway communication link. A gateway communication link among subgroups implies performance bottleneck and a single point of failure. In order to increase the throughput and reliability of inter-subgroup communication, messages are in parallel transmitted in a network striping way through multiple channels between multiple processes in the subgroups. We discuss a striping multi-channel inter-subgroup communication protocol (SMIP). We evaluate SMIP in terms of stability of bandwidth and message loss ratio and show how SMIP can support more stable bandwidth and message loss ratio.

Keywords: Distributed multimedia system, Group communication, Network striping, Hierarchical group, Large-scale group

1 Introduction

Multimedia messages are exchanged among processes in distributed applications like teleconference, video on demand (VoD), and video surveillance systems [4]. Each application requires a system to support some quality of service (QoS) like bandwidth, delay time, and packet loss ratio. It is critical to

discuss how to support each of huge number and various 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) [21] and Grid [11] computing systems, hundreds to thousands, possibly million number of peer processes are cooperating, which are widely distributed in networks, by exchanging messages with each other in networks.

Traditional communication protocols like TCP [23] and RTP [25] support processes with reliable and efficient one-to-one transmission and one-to-many transmission of messages, respectively. Here, a process can reliably and efficiently transmit messages to one or more than one process. Recently, multiple connections are used to in parallel transmit messages from a process to another process in *network striping* technologies [2, 8, 26] in order to increase the throughput. In PSockets [26], data is divided into partitions and each partition is transmitted at a different socket. SplitStream [8] is a system to distribute contents with high-bandwidth over peer-to-peer (P2P) overlay network. The multimedia content is striped and distributed using separate multicast trees with disjoint interior nodes.

In group communications, a group of peer processes are cooperating by exchanging messages while processes not only send messages to but also receive messages from multiple processes [6]. Various types of group communication protocols [6, 13, 28] are discussed to causally deliver messages [19]. The communication overhead to exchange messages is $O(n)$ to $O(n^2)$ for the number n of processes in a group. Here, every process directly sends a message to multiple destination processes while receiving messages from multiple processes in a group. In order to reduce the communication overheads, hierarchical groups are discussed where a group is divided to smaller subgroups. Each subgroup can be furthermore divided to smaller subgroups. While directly exchanging messages with the other processes in each subgroup, each process exchanges messages with other processes only through a gateway process. Takamura *et al*. [30] discuss how to support the causally ordered delivery of messages in a hierarchical group by using the vector clock. Here, a group is composed of subgroups where processes in different subgroups exchange messages via gateway processes. Taguchi *et al*. [28, 29] discuss multi-layered group protocols which adopt a vector clock whose size is the number of processes in a subgroup. In Totem [18], 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 rings. Kawanami *et al.* [15] discuss a hierarchical group where real-time clock is used to causally deliver messages. The authors [20] discuss how to design a hierarchical group from a large number of processes by using the *k*-medoid clustering algorithms [14]. Here, a hierarchical group is designed so as to minimize the average delay time between processes.

In these hierarchical protocols, a gateway process in one subgroup exchanges messages with other subgroups. Each gateway process implies not only performance bottleneck but also single point of failure since every inter-subgroup message passes the gateway. In this paper, we discuss a hierarchical group where a pair of subgroups are interconnected through multiple channels among multiple processes in the subgroups to realize parallel, reliable network striping [26]. That is, a pair of subgroups communicate with one another in the many-to-many type of communication. In addition, the number of connections among subgroups can be dynamically changed, i.e. the more number of connections are used, the higher bandwidth and reliability are supported for applications.

In section 2, we discuss a model of a hierarchical group. In section 3, we discuss inter-subgroup communication in a hierarchical group. In section 4, we discuss how to design a hierarchical group for a given set of processes distributed in networks. In section 5, we evaluate the inter-subgroup

communication protocol in terms of bandwidth and message loss ratio compared with the one-to-one communication and the hierarchical group in terms of delay time and the number of messages compared with the flat group.

2 Striping Hierarchical Group

2.1 Hierarchical group

A *group* of multiple peer processes are cooperating by exchanging messages in order to achieve some objectives. In one-to-one and one-to-many type of communications [9], each message is *reliably* routed to one or more than one process, respectively. On the other hand, a process sends a message to multiple processes while receiving messages from multiple processes in group communications [5, 6, 8, 19, 27]. 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 message m_1 *happens before* [16] a sending event of message m_2 [5]. For example, suppose a process p_1 sends a message m_1 to a pair of processes p_2 and p_3. After receiving the message m_1, the process p_2 sends a message m_2 to the processes p_3. Here, the message m_1 causally precedes the message m_2 ($m_1 \rightarrow m_2$). A common destination process p_3 of the messages m_1 and m_2 is required to deliver the message m_1 before m_2. Linear clock [16], vector clock [17], and physical clock with a GPS time server [15] are used to causally deliver messages in distributed systems. Each process gives each message m timestamp $m.T$ which shows time of the process. If $m_1 \rightarrow m_2$, $m_1.T < m_2.T$ in every clock. Furthermore, $m_1 \rightarrow m_2$ if $m_1.T < m_2.T$ in the vector clock. However, each message is required to bring a vector of elements n for number m of processes.

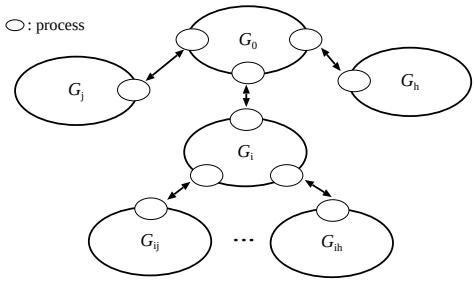

Fig. 1. Hierarchical group.

In a *flat* group, every pair of peer processes directly exchange messages with one another. Most group protocols [6, 19, 27] are discussed for flat groups with the vector clock. Due to computation and communication overheads $O(n)$ to $O(n^2)$ for the total number n of processes in a flat group with the vector clock, a large number n of processes cannot be supported. In addition, it is difficult to maintain the membership, i.e. what processes are operational in a scalable group. First, processes in a group G are partitioned into multiple subgroups. There is one *root* subgroup G_0 which is connected with subgroups $G_1, \ldots, G_k$ ($k \geq 1$). Then, each subgroup G_i is furthermore connected with subgroups $G_{i1} \ldots G_{ik_i} (k_i \geq 0)$ as shown in Figure 1. Here, a subgroup G_i is referred to as a *parent* of a *child* subgroup G_{ij}. In a hierarchical group [28], every pair of a parent subgroup G_i and a child subgroup G_{ij} communicate with one another through one gateway link as shown in Figure 3a. Hence, the gateway processes and inter-gateway communication channel imply performance bottleneck and a single point of failure.

2.2 Inter-subgroup communication

In order to increase the performance and reliability of inter-subgroup communications, we newly discuss a *Striping Multi-channel Inter-subgroup communication Protocol* (*SMIP*). Here, every pair of parent and child subgroups communicate with one another through multiple channels as shown in Figure 3b. A gateway process p_{ij} in a subgroup G_{ij} communicates with a parent subgroup G_i and child subgroup G_{ijh}. Gateway processes communicating with a parent subgroup G_i and child subgroup G_{ijh} are *upward* and *downward* gateway processes, respectively, in a subgroup G_{ij} [Figure 2]. Each process can be both types of gateways. A process is referred to as *normal* if and only if (iff) the process does not play a role of any type of gateway. A *leaf* subgroup includes normal processes and only upward gateway processes. If every leaf subgroup is at the same layer of the hierarchy, the hierarchical group is referred to as *height-balanced*.

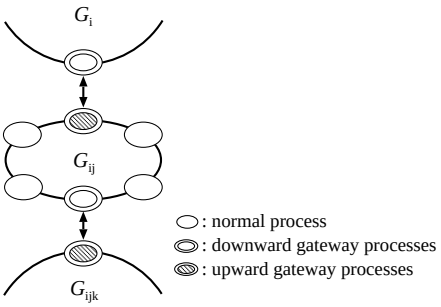

Fig. 2. Gateway processes.

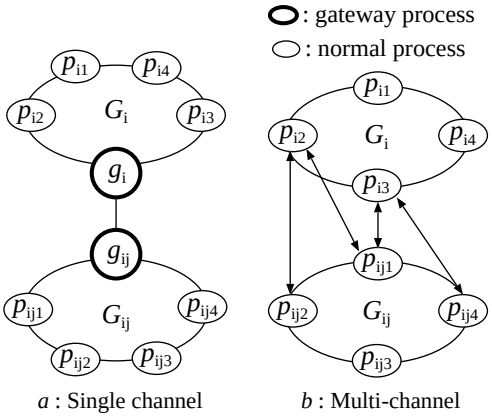

Fig. 3. Inter-subgroup communication.

The maximum size of each subgroup is bounded due to the limited computation power of each process. The number s_i of processes in a subgroup G_i has to satisfy a condition $s \leq s_i \leq S$ where constants s and S show the minimum and maximum numbers of processes, respectively, in the subgroup G_i. The smaller size of each subgroup is, the more number of subgroups are connected, i.e. the

height or breadth is increased in the hierarchy of subgroups. If the number k_i of child subgroups of the subgroup G_i is increased, the overhead for inter-group communication is increased. Processes leave and join the subgroup G_i. In addition, the quality of service (QoS) supported by a process or network is changed. Processes in a subgroup may move to another subgroup to satisfy the performance and QoS requirements. If $s_i > S$, the subgroup G_i is split into smaller subgroups. If $s_i < s$, one of the following actions :

1. The subgroup G_i is merged into a sibling subgroup G_j.
12. Processes in the subgroup G_i and its sibling subgroup G_j are redistributed into the subgroups G_i and G_j so that the subgroup G_i and G_j satisfy the condition.

A hierarchical group is dynamically height-balanced as discussed in the B-tree [3]. The authors discuss how to construct and maintain a hierarchical group from a large number of peer processes [20].

In this paper, we assume each process broadcasts every message m to all the processes in a group as follows :

1. Each process sends a message m to every process in a local subgroup G_{ij}, normal processes and upward and downward gateway processes.
2. An upward gateway process forwards the messages m up to downward gateway processes of the parent subgroup G_i.
3. A downward gateway process forwards the messages 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 [17] and linear clock [16]. Even if a message m causally precedes another message m_2 in a local subgroup, the messages m_1 and m_2 may be causally concurrent in a whole group. In the paper [28], it is discussed how to resolve the unnecessary ordering of messages in a hierarchical group.

3 Striping Inter-subgroup Communication

3.1 Inter-subgroup communication

In order to increase the performance and reliability of the inter-subgroup communication, a pair of parent and child subgroups G_i and G_{ij} communicate with one another through multiple channels with multiple gateway processes. That is, a pair of subgroups communicate with one other in a the many-to-many type of communication among gateway processes. Here, let us consider a subgroup G_i and its child subgroup G_{ij}. Downward gateway processes in a subgroup G_i are communicating with upward gateway processes in a child subgroup G_{ij} in the many-to-many communication as shown in Figure 3b.

Suppose gateway processes in a subgroup G_i send messages to gateway processes in another subgroup G_{ij}. A gateway process which sends a message to another gateway process is referred to as a *source* gateway processes of the message. On the other hand, a gateway process which receives a message from another gateway process is referred to as *destination* gateway processes. In our sub-group communication model, multiple gateway process in a subgroup G_i forward messages to gateway processes in another subgroup G_j There are following ways for source gateway processes to send messages to destination gateway processes in a subgroup G_j :

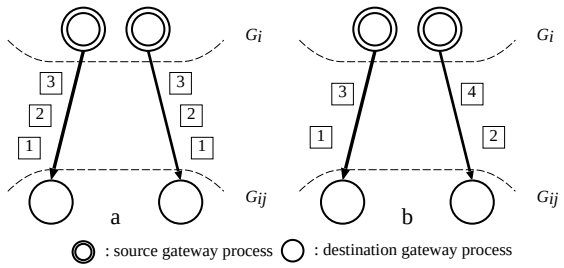

Fig. 4. Communication model of multiple source gateway processes.

a. Each source gateway process sends same messages to each of the destination gateway processes [Figure 4a].
b. Each source gateway process sends messages different from the other gateway processes to each of the destination gateway processes [Figure 4b].

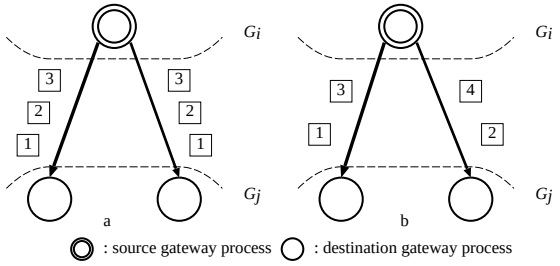

Fig. 5. Communication model of single source gateway process.

In addition, to each source gateway process in a subgroup G_i transmits messages to multiple destination gateway processes in the subgroup G_j. There are following ways for each source gateway process to transmit messages to destination processes :

a. A source gateway process in G_j transmits same messages to each of the destination gateway processes [Figure 5a].
b. A source gateway process in G_j transmits different messages to each of the destination gateway processes [Figure 5b].

If different messages are transmitted in different channels [Figures 4b and 5b], messages arrive at a destination process out of order. The destination process has to first buffer messages on receipt of the messages. Then, the messages are reordered in the sending order by using the sequence numbers of the messages. It takes time to reorder messages in the buffer since a process has to wait for delayed messages. We have to reduce the number of messages to be reordered to increase the performance. We would like to discuss this reordering problem in another paper.

3.2 Striping multi-channel communication

Suppose a gateway process in a subgroup G_i would like to send messages to gateway processes in another subgroup G_j. In this paper, we take the following inter-subgroup transmission protocol from a source subgroup G_i to another destination subgroup G_j :

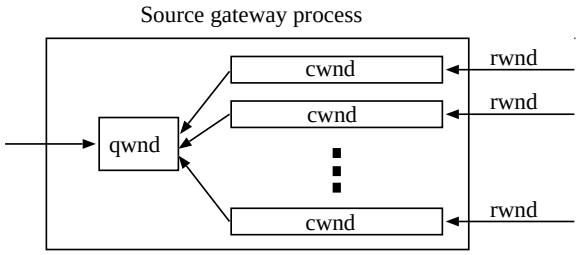

Fig. 6. Three window parameter of SMIP.

1. One process p_{is} is taken as a source gateway process in the source subgroup G_i.
2. On receipt of a message in the source subgroup G_i, the gateway process p_{is} forwards the message to some process, say p_{jt_j} in the destination subgroup G_j. Here, p_{jt_j} is a destination process of the destination subgroup G_j.
3. On recept of messages, the process p_{jt_s} forwards the messages to the destination gateway processes in the destination subgroup G_j.
4. If the channel between a pair of gateway processes p_{is} and p_{jt_1} might not support enough QoS, the source gateway process p_{is} takes another process p_{jt_2} as a gateway process in the destination subgroup G_j.
5. Thus, the source gateway process p_{is} in the source subgroup G_i sends different messages to a pair of the destination gateway processes p_{jt_1} and p_{jt_2} in the destination gateway subgroup G_j. The process p_{is} distributes messages to a pair of the gateway processes p_{jt_1} and p_{jt_2} so that both the channels with the processes p_{jt_1} and p_{jt_2} satisfy the QoS requirement.
6. The larger bandwidth is required, the more number of destination gateway processes are taken in the destination subgroup G_j. The source gateway process p_{is} sends messages to the destination gateways in the source subgroup G_i.

Messages are transmitted in a channel between a pair of gateway processes by the congestion control algorithm used in TCP [12]. If a pair of subgroups are interconnected in a single channel, processes in the subgroups cannot communicate with each other due to the congestion and fault of the channel. In the inter-subgroup communication protocol SMIP, a pair of subgroups G_i and G_j are interconnected with many-to-many types of communication channels. Even if a channel is faulty or does not support QoS requirement, the subgroups can communicate with one another with enough QoS through other operational channels. The approach has the following advantages :

1. The network traffic can be distributed to multiple channels.
2. The other channels compensate the degradation of QoS even if QoS of some channel is degraded.

Messages are transmitted in each channel between a pair of source and destination gateway processes through the congestion control algorithm, the *additive increase and multiplicative decrease* (*AIMD*) algorithm used in TCP [12]. Here, two parameters, *congestion window size* ($cwnd$) and *receiver window size* ($rwnd$) are used for each channel. In our protocol, an additional parameter *requirement window* ($qwnd$) showing the size of data in the buffer is used for a set of the channels as shown in Figure 6. The window size (wnd) of each channel is decided as follows :

$$wnd = min(cwnd, rwnd, qwnd).$$

The source gateway process p_{is} in a subgroup G_i sends messages to a destination gateway process p_{jt_l} in another subgroup G_j through a channel. The requirement window size ($qwnd$) is decided as follows :

$$qwnd = qwnd - wnd.$$

4 Design of Hierarchical Group

We discuss how to design a hierarchical group for a set $\mathbf{G}$ of peer processes $p_1, \ldots, p_n$. 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 [22] or a connection of TCP [23]. Each channel C_{ij} is characterized by 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 networks. In order to realize real-time multimedia communications, it is critical to reduce 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 among processes.

Let d_{ij} stand for the message delay time from a process p_i to another process p_j. 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 the processes p_i and p_j. The distance is assumed to be symmetric from the destination in this paper. Let $\mathbf{D_G}$ be a set of distances between every pair of processes in a set $\mathbf{G}$ of processes, $\{\delta(p_i, p_j) |\ p_i, p_j \in \mathbf{G}\}$. ***AvDist***$(p_i, \mathbf{G})$ shows the average distance from a process p_i to every other process in the process set $\mathbf{G}$, i.e. $\sum_{p_j \in \mathbf{G}} \delta(p_i, p_j)/(|\mathbf{G}| - 1)$.

Given a process set $\mathbf{G}$ and the distance set $\mathbf{D_G}$ for $\mathbf{G}$, a parent subgroup $\mathbf{G}_0$ and child subgroups $\mathbf{G}_1, \ldots, \mathbf{G}_k$ are obtained by the following procedure ***DV*** where s is the number of processes to be included in the parent subgroup $\mathbf{G_0}$ and k is the number of child subgroups of the parent subgroup $\mathbf{G_0}$. Here, $\mathbf{G} = \mathbf{G}_0 \cup \mathbf{G}_1 \cup \ldots \cup \mathbf{G}_k, \mathbf{G}_0 \cap \mathbf{G}_i = \phi$, and $\mathbf{G}_i \cap \mathbf{G}_j = \phi (for i.j = 1, dots, k, i \neq j)$.

DV$(\mathbf{G}, \mathbf{D_G}, s, k)$ {
  $\mathbf{G}_0 :=$ ***Parent***$(\mathbf{G}, \mathbf{D_G}, s)$;
  $\{\mathbf{G}_1, \ldots, \mathbf{G}_k\} :=$ ***Child***$(\mathbf{G} - \mathbf{G}_0, \mathbf{D}_{\mathbf{G}-\mathbf{G}_0}, k)$;
  if $\mathbf{G} = \mathbf{G}_0$,
    for $i = 1, \ldots, k, \{$
      $\mathbf{G}_{i0} :=$ ***DV***$(\mathbf{G}_i, \mathbf{D_G}, s)$;
      Let $\mathbf{G}_{i0}$ be a child of $\mathbf{G}_0$; $\}$
  return$(\mathbf{G}_0)$;
}

First, a parent subgroup $\mathbf{G}_0$ is obtained by the procedure ***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 ***Parent*** is given as follows :

Parent$(\mathbf{G}, \mathbf{D_G}, s)$ {
  $\mathbf{G}_0 := \phi$; $i := 0$;
  while$(i < s)$ {
    Select a process p whose ***AvDist***$(p, \mathbf{G})$ is the minimum in $\mathbf{G}$.

if $i < s/2$ **or** ***AvDist***$(p, \boldsymbol{G}) <$ ***AvDist***$(p', \boldsymbol{G}_0) + \alpha$ where p' is a process
whose ***AvDist***$(p', \boldsymbol{G}_0)$ is the minimum in $\mathbf{G}_0$,
$\{\mathbf{G}_0 := \mathbf{G}_0 \cup \{p\}$; $\mathbf{G} := \mathbf{G} - \{p\}$; $i := i + 1$;$\}$
else return ($\mathbf{G}_0$);
}
return ($\mathbf{G}_0$);
}

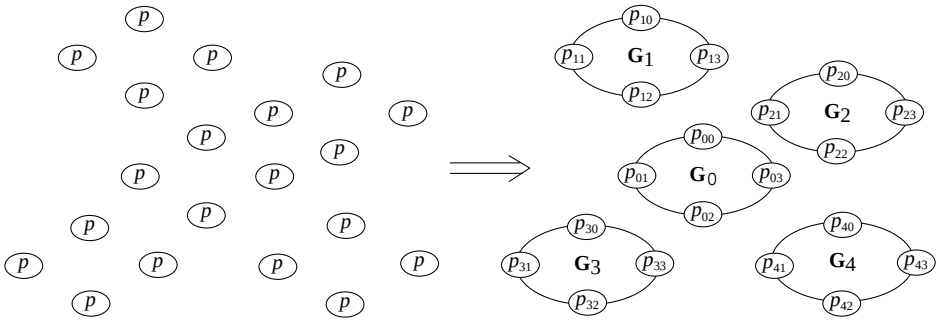

Fig. 7. ***k***-partitioning of a group.

Here, α is a constraint. The larger the constant α is, the more distant processes are included in a subgroup. By executing the procedure ***Parent***, a parent subgroup $\mathbf{G}_0$ is obtained and processes in the parent subgroup $\mathbf{G}_0$ are removed from the process set $\mathbf{G}$. By the ***Child*** procedure using a type of *k*-medoids algorithm [14], a group $\mathbf{G}$ is partitioned into k child subgroups $\mathbf{G}_1, \ldots \mathbf{G}_k$. In Figure 7, a root subgroup $\mathbf{G}_0$ of four processes is first obtained by ***Parent***. Then, remaining 16 processes are partitioned into four subgroups $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$, and $\mathbf{G}_4$, each of which includes four processes. Processes which are nearer to each other in a group $\mathbf{G}$ are grouped into one subgroup. That is, $\delta(p_i, p_k) < \delta(p_i, p_j)$ for every pair of processes p_i and p_k in a subgroup $\mathbf{G}_i$ and every process p_j in another subgroup $\mathbf{G}_j (j \neq i)$. There are algorithms like PAM (Partitioning Around Medoids) [14] and CLARA (Clustering LARge Applications) [14] to partition a collection of data into clusters. PAM is efficient for smaller number of processes ($n < 100$) and CLARA can be adopted for more number of processes.

Algorithm PAM

1. Select k representative processes arbitrarily in $\mathbf{G}$.
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 $TC_{ij} < 0$, the process p_j gets selected and the other process p_i is changed to a non-selected one. Goto 2.
4. Otherwise, for each non-selected process p_j, find the most nearer selected process p_i and include the process p_j to a subgroup of the process p_i. Halt.

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, i.e. selected process which is to be replaced. The following processes are defined.

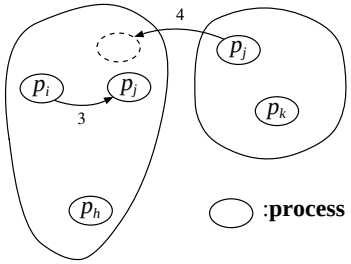

Fig. 8. Replacement of a medoid.

$p_h =$ the new medoid with which p_i is replaced.

$p_j =$ another non-selected process which may or may not need to be changed in the subgroup

$p_k =$ a current medoid which is nearest to p_j.

The total cost TC_{ih} for a pair of the processes p_j and p_h is given $\sum_j C_{jih}$. Here, C_{jih} is computed as follows [Figure 8] :

1. Suppose a process p_j is in a subgroup of a selected process p_i and a process p_h is the second closest 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) $C_{jih} = \delta(p_j, p_h) - \delta(p_j, p_i)$ otherwise.
2. Suppose a process p_j currently belongs to a subgroup other than the one represented by p_h. Let p_k be the selected process of that subgroup.
   (a) $C_{jih} = 0$ if $\delta(p_j, p_k) > \delta(p_j, p_h)$.
   (b) $C_{jih} = \delta(p_j, p_h) - \delta(p_j, p_k)$ otherwise.

Next, the algorithm CLARA is shown as follows :

Algorithm CLARA

1. For $i := 1$ to N, repeat the following steps:
2. Arbitrarily select a sample set **S** of S 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 **G**, determine which of the k medoids is the most nearer to the process p_j and add the process p_j to the subgroup of the medoid.
4. Calculate the average distance δ of the subgroup obtained in the previous step. If δ is smaller 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.

In CLARA, $N = 5$ and $S = 40 + 2k$. The complexity of a single iteration is $O(k(n-k)^2)$ in PAM and $O(kS^2 + k(n-k))$ 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.

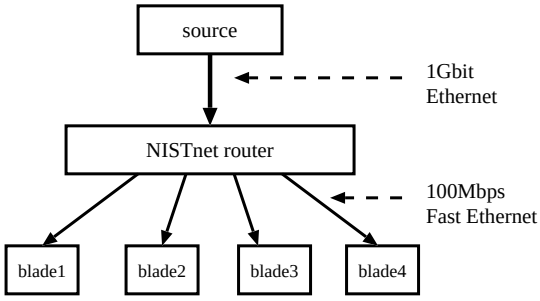

Fig. 9. Data transfer arrangement for SMIP.

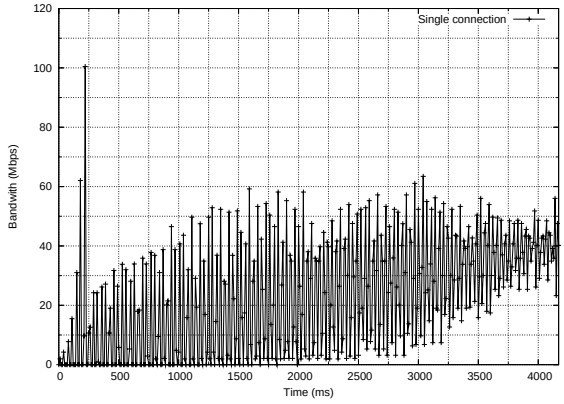

Fig. 10. Bandwidth adaptation on traditional one-channel model.

5 Evaluation

5.1 Evaluation of inter-subgroup communication protocol

We evaluate the striping multi-channel inter-subgroup communication protocol (SMIP) in terms of the stability of bandwidth and the message loss ratio compared with the traditional one-channel transmission protocol like TCP. In the traditional one-to-one communication approach, protocols like RSVP [24] at a lower layer than the transport layer are used to support the quality of service (QoS) required by applications. In our striping multi-channel approach, QoS is supported on the end-to-end basis with QoS control at transport layer. In the simulation, the bandwidth of the network channel is bounded to be 30Mbps by the evaluation tool although the channel support larger bandwidth 30Mbps means the transmission speed of the digital video (DV) data.

Figure 9 shows the evaluation environment of the striping multi-channel communication protocol. A source gateway process is realized in a computer Dell Precision 530 with dual Intel Pentium Xeon 1.8Ghz and 1.5B memory on Linux 2.6.10. Four destination gateway processes are realized in HP Proliant BL10e blade server with Intel PentiumM 1Ghz and 512MB memory on Linux 2.4.26. These gateway processes are interconnected through a computer HP Proliant DL145 with dual AMD Opteron 2.2Ghz and 2GB memory on Linux 2.4.21 named NISTnet router where NISTnet [7] is installed. The delay time between a source gateway process and a destination gateway process is emulated to be 40 milliseconds by using the NISTnet.

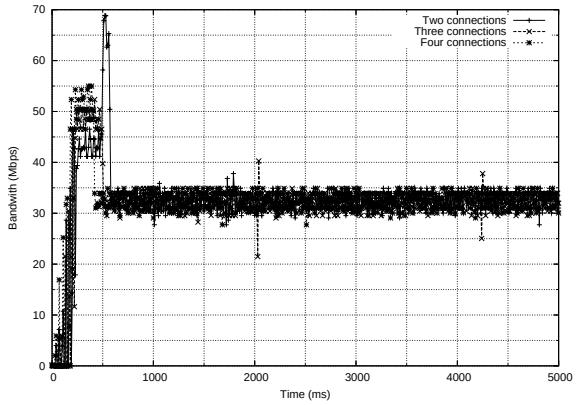

Fig. 11. Bandwidth adaptation on striping multi-channel model.

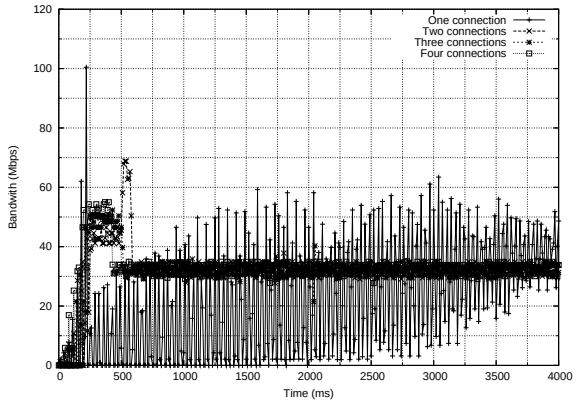

Fig. 12. Bandwidth adaptation.

In the evaluation, the source gateway process sends multimedia like DV data with 30Mbps. The NewReno algorithm [10] of TCP is used for transmitting messages in each channel. The data transmission procedure of TCP is emulated over UDP/IP [22]. Figure 10 shows how the bandwidth is changed for time in the traditional one-channel transmission. The bandwidth supported is largely changed. Figure 11 shows the bandwidth in our striping multi-channel transmission. Compared with the one-channel transmission, the striping multi-channel transmission supports more stable bandwidth, i.e. 30Mbps. The DV data is required to be transmitted, i.e. the constant bandwidth 30Mbps. In the SMIP, the bandwidth of 30Mbps can be continually supported. However, the bandwidth supported by the traditional one-channel protocol is not so stable that the DV data cannot be transmitted. Figure 12 shows both the one-channel and the striping multi-channel ways to show how stable the striping multi-channel way is. Even if QoS is degraded in a channel, messages which cannot be transmitted in the channel can be transmitted through the other channels in the striping multi-channel approach.

Next, we measure the message loss ratio. We take a pair of subgroups G_i and G_j. In the traditional way, one gateway process in the subgroup G_i communicates with one gateway process in the other subgroup G_j [Figure 13a]. In the SMIP, the same number k of gateway processes in each of the subgroup G_i and G_j are interconnected. Here, this inter-subgroup communication from l gateway

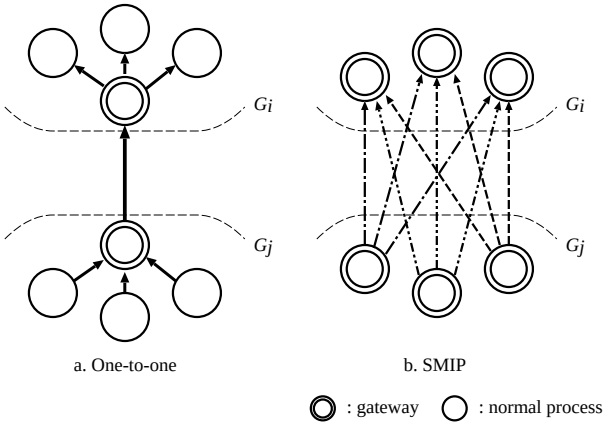

Fig. 13. Data transfer arrangement.

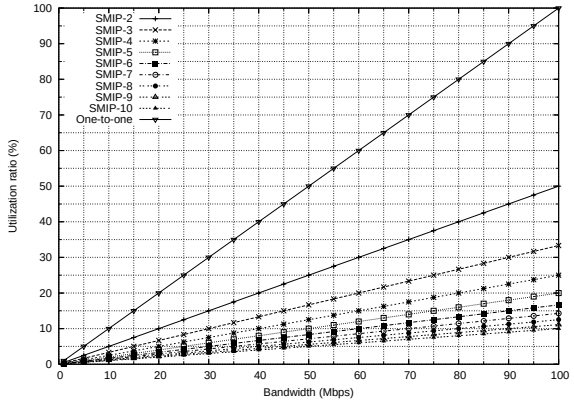

Fig. 14. Utilization of bandwidth.

processes to l gateway processes is written in SMIP-l. Figure 13b shows SMIP-3. Each pair of gateway processes are interconnected in the 100Mbps Fast Ethernet. Each of normal processes and gateway processes is realized in an HP Proliant BL10e blade server with Intel PentiumM 1Ghz and 512MB memory on Linux 2.4.26. Gateway processes are interconnected through a computer HP Proliant DL145 with dual AMD Opteron 2.2Ghz and 2GB memory on Linux 2.4.21 named NISTnet router where NISTnet [7] is installed. The delay time a pair of the subgroups between G_i and G_j is emulated to be 40 milliseconds by using the NISTnet. Figure 15 shows the packet loss ratio for the bandwidth for each gateway process for the traditional one-to-one and SMIP-3. In the SMIP, no packet is lost. In Figure 15, k [Mbps] means the each of three gateway processes sends packets with $k/3$ [Mbps]. On the other hand, the message loss ratio is increased as the transmission bandwidth of each gateway process is increased. For example, about 0.018% of packets transmitted are lost if a gateway process transmits messages with 60Mbps. Figure 14 shows that the utilization of the bandwidth of each gateway process. In the traditional way, the bandwidth is used out. This means, the transmission rate cannot be increased. On the other hand, messages are transmitted through multiple channels in SMIP. Hence, each gateway can have unused bandwidth to transmit further messages in

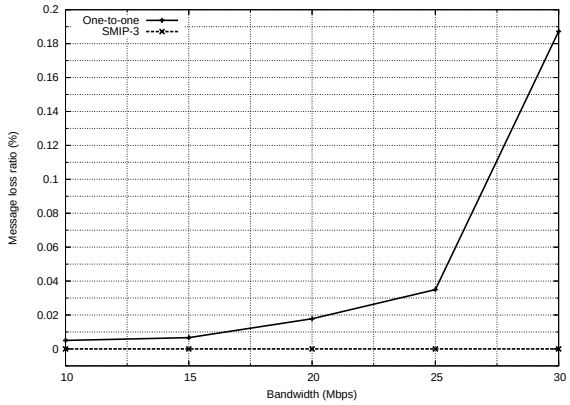

Fig. 15. Message loss ratio.

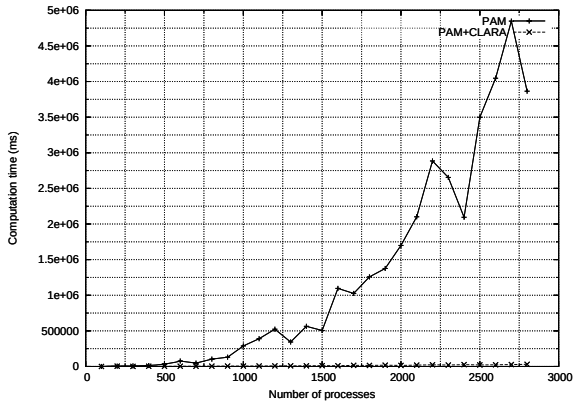

Fig. 16. Performance of the PAM and PAM+CLARA.

SMIP.

5.2 Evaluation of designing hierarchical group

We implement three versions $\boldsymbol{DV}_p$, $\boldsymbol{DV}_c$, and $\boldsymbol{DV}_{cp}$ of the procedure $\boldsymbol{DV}(\mathbf{G}, \mathbf{D_G}, s, k)$ which take usage of the PAM, CLARA, and PAM+CLARA algorithms, respectively. Let n be the number of processes in a group **G**. In the PAM+CLARA algorithm, CLARA is a adopted for $n \geq 200$ and PAM for $n \leq 200$. First, we measure how long it takes to obtain a hierarchical group $\mathbf{H_G}$ for a group **G** of n processes. Figures 16 and 17 show the computation time to make a hierarchical group $\mathbf{H_G}$ for a group **G** of n processes by three algorithms $\boldsymbol{DV}_p$, $\boldsymbol{DV}_c$, and $\boldsymbol{DV}_{cp}$. $\boldsymbol{DV}_{cp}$ is the fastest for every number n of processes as shown in Figures 16 and 17. Hence, we take the algorithm $\boldsymbol{DV}_{cp}$ to obtain a hierarchical group $\mathbf{H_G}$.

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 a 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 among a pair of neighboring nodes in the lattice shows a distance of one

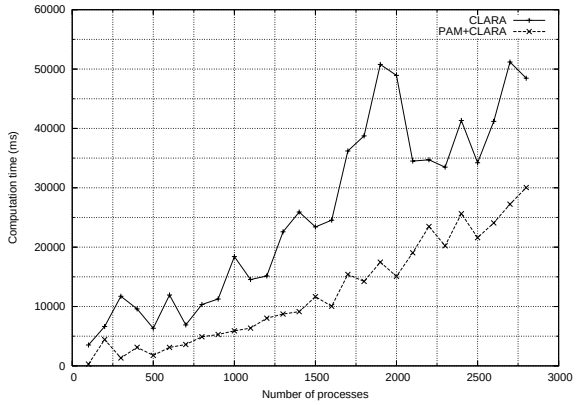

Fig. 17. Performance of the CLARA and PAM+CLARA.

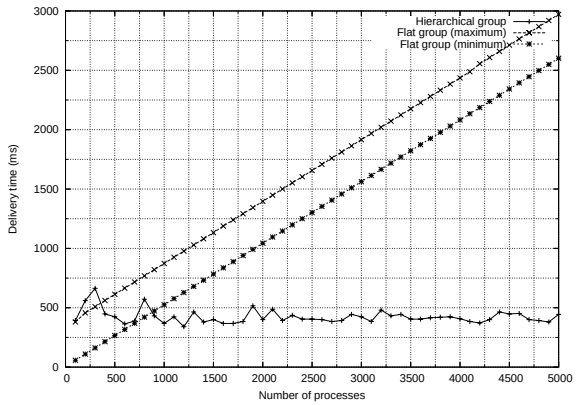

Fig. 18. Delivery time.

milisecond 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 the processes p_i and p_j. The number s of processes in each subgroup is decided for the total number n of processes; $s = n/10$, if $n \leq 500$ and $s = 50$, otherwise.

The height h of the hierarchical group $\mathbf{H_G}$ is decided for n; $h = 10 + \lfloor n/500 \rfloor$.

The number k of child subgroups is computed to be $\lfloor (n/s)^{1/h-1} \rfloor$. In the flat group **G**, a process directly sends a message $(n-1)$ times to deliver to $(n-1)$ processes. For example, it takes 52 [msec] for a process to transmit 100 messages in Linux 2.4.26 on a personal computer Dell Precision 530 with dual Intel Pentium Xeon 1.8Ghz and 512MB memory. 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. In the hierarchical group $\mathbf{H_G}$, the delivery time of a message from a process to the most distant process is measured, which is the maximum one. That is, a message sent by a process is forwarded via a root subgroup to a destination process. Figure 18 shows the maximum and minimum delivery time in the flat group **G** and the maximum delivery time in the hierarchical group $\mathbf{H_G}$. The maximum delivery time of the hierarchical group $\mathbf{H_G}$ is almost constant while the

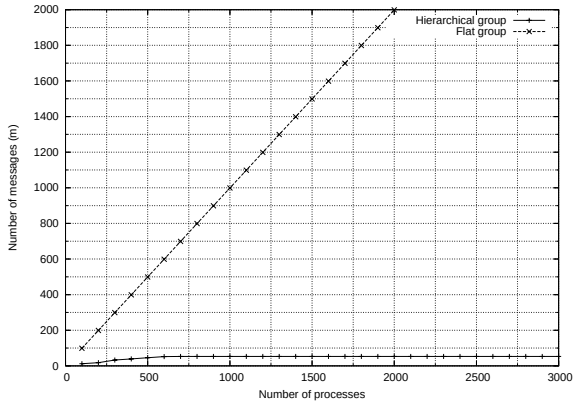

Fig. 19. Number of messages.

delivery time is $O(n)$ in the flat group **G**.

We measure how many messages are transmitted. In the flat group **G**, a process sends $(n-1)$ messages to broadcast to all the processes. On the other hand, in the hierarchical group $\mathbf{H_G}$, a message is broadcast in each subgroup. Figure 19 shows the number of processes transmitted in the flat group **G** and the maximum number of messages transmitted in the hierarchical group $\mathbf{H_G}$. In the hierarchical group $\mathbf{H_G}$, the number of messages transmitted can be drastically reduced.

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 subgroups are interconnected through multiple communication channels between multiple gateway processes in the subgroup. First, we discussed algorithms to design a hierarchical group for a large number of processes distributed in a network so as to minimize the average delay time. In traditional hierarchical groups, a pair of subgroups communicate with one another through a pair of the gateway processes. The communications through the gateways implies performance bottleneck and single point of failure. Gateway processes in different subgroups exchange messages through multiple channels with multiple gateway processes in the network striping way to increase the reliability and performance of the inter-subgroup communication. In the evaluation, we showed that the hierarchical group supports the shorter delay time and the fewer number of messages than the flat group. In addition, we showed that the striping multi-channel inter-subgroup communication protocol can support the higher stability of the bandwidth and the smaller message loss ratio compared with the traditional protocol.

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. M. Beynon, D. Van Hook, M. Seibert, A. Peacock, and D. Dudgeon. Detecting abandoned packages in a multi-camera video surveillance system. In *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS 2005)*, pages 221–228, 2003.
5. K. Birman. Lightweight Causal and Atomic Group Multicast. *ACM Trans. on Computer Systems*, pages 272–290, 1991.
6. K. P. Birman and R. v. Renesse. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1994.
7. M. Carson and D. Santay. NIST Net: a Linux-based Network Emulation Tool. *Computer Communication Review*, 33(3):111–126, 2003.
8. 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.
9. S. Deering. Host Groups: A Multicast Extension to the Internet Protocol. *RFC 966*, 1985.
10. S. Floyd, ICSI, T. Henderson, Boeing, A. Gurtov, and TeliaSonera. The NewReno Modification to TCP's Fast Recovery Algorithm. *Request for Comments 3782*, 2004.
11. I. Foster and C. Kesselman. *The Grid2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 2003.
12. V. Jacobson. Congestion Avoidance and Control. In *Proc. of the ACM Symposium on Communications Architectures and Protocols (SIGCOMM '88)*, pages 314–329, 1988.
13. M. F. Kaashoek and A. S. Tanenbaum. An Evaluation of the Amoeba Group Communication System. In *Proc. of the IEEE 16th International Conference on Distributed Computing Systems (ICDCS '96)*, pages 436–447, 1996.
14. L. Kaufman and P. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley & Sons, 1990.
15. S. Kawanami, T. Enokido, and M. Takizawa. Heterogeneous Groups to Causally Ordered Delivery. In *Proc. of the IEEE 6th International Workshop on Multimedia Network Systems and Applications (MNSA 2004)*, pages 70–75, 2004.
16. L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, 1978.
17. F. Mattern. Virtual Time and Global States of Distributed Systems. *Proc. of the International Workshop on Parallel and Distributed Algorithms*, pages 215–226, 1989.
18. 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.
19. A. Nakamura and M. Takizawa. Causally Ordering Broadcast Protocol. In *Proc. of the IEEE 14th International Conference on Distributed Computing Systems (ICDCS '94)*, pages 48–55, 1994.
20. Y. Nishimura, T. Enokido, and M. Takizawa. Design of a Hierarchical Group to Realize a Scalable Group. In *Proc. of the IEEE 19th International Conference on Advanced Information Networking and Applications (AINA 2005)*, pages 9–14, 2005.
21. A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.
22. J. Postel. User Datagram Protocol. *Request for Comments 768*, 1980.
23. J. Postel. Transmission Control Protocol. *Request for Comments 0793*, 1992.
24. E. R. Braden, ISI, L. Zhang, UCLA, S. Berson, ISI, S. Herzog, IBM Research, S. Jamin, and Univ. of Michigan. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. *Request for Comments 2205*, 1997.
25. H. Schulzrinne, R. Casner, S. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-time Applications. *Request for Comments 1889*, 1996.
26. 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.
27. T. Tachikawa, H. Higaki, and M. Takizawa. Group Communication Protocol for Realtime Applications. In *Proc. of the IEEE 18th International Conference on Distributed Computing Systems (ICDCS 98)*, pages 40–

47, 1998.

28. K. Taguchi, T. Enokido, and M. Takizawa. Causal Ordered Delivery for a Hierarchical Group. In *Proc. of the IEEE 10th International Conference on Parallel and Distributed Systems (ICPADS 2004)*, pages 485–492, 2004.
29. K. Taguchi and M. Takizawa. Two-Layered Protocol for a Large-Scale Group of Processes. In *Proc. of the IEEE 9th International Conference on Parallel and Distributed Systems (ICPADS 2002)*, pages 171–176, 2002.
30. M. Takizawa, M. Takamura, and A. Nakamura. Group Communication Protocol for Large Group. In *Proc. of the IEEE Conference on Local Computer Networks (LCN '93)*, pages 310–319, 1993.