SMIP: Striping Multimedia Communication Protocol for Large Scale Hierarchical Group

Yasutaka Nishimura[1], Naohiro Hayashibara[1], Tomoya Enokido[2], and Makoto Takizawa[1]
[1]*Dept. of Computers and Systems Engineering, Tokyo Denki University, Japan*
[2]*Faculty of Business Administration, Rissho University, Japan*
[1]{*yasu, haya, taki*}*@takilab.k.dendai.ac.jp,* [2]*eno@ris.ac.jp*

Abstract

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 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.

1. Introduction

Multimedia messages are exchanged among peer processes in distributed applications like teleconferences, video on demand, and video surveillance systems in peer-to-peer (P2P) overlay network [12]. Each application requires a system to support quality of service (QoS) like bandwidth, delay time, and packet loss ratio.

Traditional communication protocols like TCP [5] support processes with reliable and efficient one-to-one and one-to-many transmission of messages. Recently, multiple connections are used to in parallel transmit messages from a process to another process in *network striping* technologies like SplitStream [3] in order to increase the throughput. SplitStream [3] is a system to distribute contents with high-bandwidth over peer-to-peer (P2P) overlay network. The multimedia content is striped and distributed using separated 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 [1]. Various types of group communication protocols [1, 14] are discussed to causally deliver messages. The communication overhead 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, a gossiping protocols [7] are discussed. Here, each process forwards a message to processes randomly selected. Even if membership is changed, the message can be eventually delivered to all the processes. However, the delivery time to all the processes cannot be fixed. In another hierarchical group approach, a group is divided to smaller subgroups. Each process exchanges messages with processes in other subgroups only through one gateway process. Taguchi *et al.* [14] discuss multi-layered group protocols which adopt a vector clock whose size is the number of processes in a subgroup. In Totem [10], 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. The authors [11] discuss how to design a hierarchical group from a large number of processes by using the k-medoid clustering algorithms [6] 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 [3].

In section 2, we discuss a model of a hierarchical group. In section 3, we discuss the striping inter-subgroup communication protocol. In section 4, we evaluate the inter-subgroup communication protocol in terms of bandwidth, message loss ratio and delay time compared with the one-to-one communication.

2. Hierarchical Group

A *group* of multiple peers are cooperating by exchanging messages in order to achieve some objectives. In the one-to-many communication, each message is *reliably* routed to one or more than one process. On the other hand, a process sends a message to multiple processes while receiving messages from multiple processes in group communications [1,3]. 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* [8] a sending event of message m_2. If p_2 sends m_2 to p_3 after receiving m_1, m_1 causally precedes

m_2 ($m_1 \rightarrow m_2$). A common destination process p_3 of m_1 and m_2 is required to deliver m_1 before m_2. Linear clock [8] and vector clock [9] are used to causally deliver messages in distributed systems.

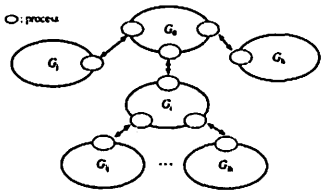

Figure 1. Hierarchical group.

In a *flat* group, every pair of peer processes directly exchange messages. Most group protocols [1] 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. 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 [14], 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 2a.

3. Striping Inter-subgroup Communication

3.1. 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 2b. 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 G_i and child G_{ijh} are *upward* and *downward* gateway processes, respectively, in G_{ij} [Figure 2]. Each process can be both types of gateways. A process is referred to as *normal* iff the process does not play a role of gateway. In this paper, we assume each process broadcasts every message to all the processes in a group :

1. Each process sends a message m to every process in a local subgroup G_{ij}.
2. An upward gateway forwards m to downward gateways of parent subgroup G_i.
3. A downward gateway forwards m to upward gateways 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 [9] and linear clock [8]. In the

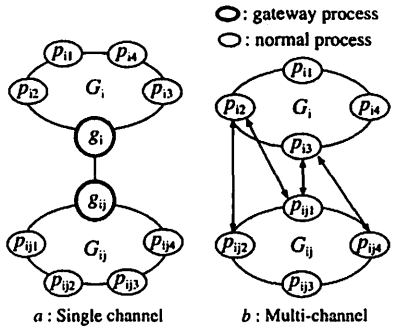

Figure 2. Inter-subgroup communication.

paper [14], it is discussed how to resolve the unnecessary ordering of messages in a hierarchical group.

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 through multiple channels with multiple gateways. That is, a pair of subgroups communicate in the many-to-many type of communication among gateways. Here, let us consider a subgroup G_i and its child subgroup G_{ij}. Downward gateways in G_i are communicating with upward gateways in a child G_{ij} in the many-to-many communication as shown in Figure 2b.

Suppose gateways in G_i send messages to gateways in another G_{ij}. A gateway which sends a message to another gateway is a *source* gateway of the message. On the other hand, a gateway which receives a message from another gateway is a *destination* gateway. In our approach, multiple gateways in G_i forward messages to gateways in G_j

3.2. Striping multi-channel communication

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

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

Messages are transmitted in a channel between a pair of gateway processes by the congestion control algo-

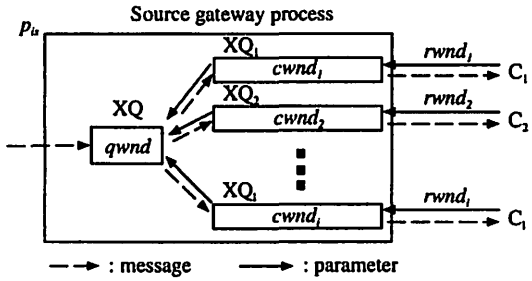

Figure 3. Three window parameter of SMIP.

rithm used in TCP [5]. In SMIP, a pair of subgroups G_i and G_j are interconnected with many-to-many types of channels. Even if a channel is faulty or does not support QoS requirement, G_i and G_j can communicate with enough QoS through other operational channels. Here, the network traffic can be distributed to multiple channels and the other channels compensate the degradation of QoS of some channel.

Messages are transmitted in each channel c_k between a pair of source and destination gateways through the congestion control algorithm, the *additive increase and multiplicative decrease* (*AIMD*) algorithm used in TCP [5]. Here, two parameters, *congestion window size* ($cwnd_k$) and *receiver window size* ($rwnd_k$) are used for each channel c_k.

The source gateway p_{is} in a subgroup G_i sends messages to a destination gateway p_{jt_k} in another subgroup G_j through a channel $c_k (k = 1,\ldots,l)$. In each channel c_k to a gateway process p_{jt_k}, the gateway process p_{is} sends messages to p_{jt_k} according to the congestion control algorithm [13]. The algorithm is composed of the following two phases :

1. Slow start phase : p_{is} sends messages by exponentially increasing the transmission rate. Here, a variable *ssThresh* shows the rate at which message loss occurs.
2. Congestion avoidance phase : p_{is} sends messages to p_{jt_k} initially at transmission rate $ssThresh/2$ and then by linearly increasing the transmission rate.

In our protocol, each gateway process transmits messages by using two types of queues, an application transmission queue XQ and local transmission queues $XQ_1,\ldots,XQ_l$ as shown in Figure 3. The parameter *qwnd* shows the *requirement window size*, i.e. the number of messages in the application transmission queue XQ.

Each destination gateway process p_{jt_k} notifies the source gateway p_{is} of the parameter $rwnd_k$ which shows the *receiver windows size*, i.e. the number of messages which p_{jt_k} can receive.

The variable $cwnd_k$ shows *congestion window size*, i.e. the number of messages to be transmitted through the channel c_k. Initially, $cwnd_k = 0$ in the slow start phase. $cwnd_k$ is incremented by one, i.e. $cwnd_k = 1$. The source gateway p_{is} sends one packet in XQ_k to p_{jt_k} through the channel c_k and waits for receipt of an acknowledgment message from p_{jt_k}. On receipt of an acknowledgment message, $cwnd_k$ is incremented by one and $wnd_k = min(cwnd_k, rwnd_k, qwnd)$. The variable wnd_k gives the number of messages which p_{is} can send to p_{jt_k}. The number wnd_k of messages are moved to the local queue XQ_k. Then p_{is} sends the number wnd_k of messages to p_{jt_k}. Thus, the transmission rate is exponentially increased. Eventually, messages are lost due to the buffer overrun. If p_{is} detects message loss on receipt of the acknowledgment message, $cwnd_k$ is decremented to be $ssThresh/2$. Then, the congestion avoidance phase is started. If p_{is} detects packet loss by the timeout mechanism, $cwnd_k = 1$ and the slow start phase is restarted.

In the congestion avoidance phase, p_{is} sends the number $cwnd_k$ ($= ssThresh/2$) of messages. On receipt of an acknowledgment message, $cwnd_k := 1/cwnd_k$ and $wnd = min(cwnd_k, rwnd_k, qwnd)$. Then, p_{is} sends the number wnd_k of messages to p_{jt_k}. If message loss is detected, the transmission rate is decreased as presented in the slow start phase.

In our protocol, the source gateway p_{is} sends in parallel messages through multiple channels. If the traditional congestion control algorithm is adopted for each channel, message loss occurs in each channel since the transmission rate is monotonically increased while no message loss in another channel. In SMIP, the transmission rate of each channel is more slowly increased to reduce the message loss. The application process puts messages to the application queue XQ at the rate of the application, e.g. 30Mbps for video transmission. *qwnd* is incremented each time a message is enqueued into XQ. Thus, *qwnd* shows the application rate. Messages in XQ are distributed to the local transmission queues $XQ_1,\ldots,XQ_l$. In our protocol, the top wnd_k messages in the application queue XQ are atomically moved to each local transmission queue XQ_k if XQ_k is empty. Thus, the messages in XQ are arbitrarily distributed to the local queue $XQ_1,\ldots,XQ_l$. For each channel c_k, messages are transmitted as follows :

1. On receipt of a packet t from an application :
   the packet t is enqueued into XQ;
   $qwnd = qwnd + 1$;
2. Slow start phase at a channel c_k:
   Initially, $cwnd_k := 1$;
   $wnd_k := min(cwnd_k, rwnd_k, qwnd)$;
   $qwnd := qwnd - wnd_k$;
   The number wnd_k of messages in XQ
   are moved to XQ_k.
   send messages from XQ_k
   On receipt of an acknowledgment message,
   if no message is lost,
   if $qwnd = 0$,

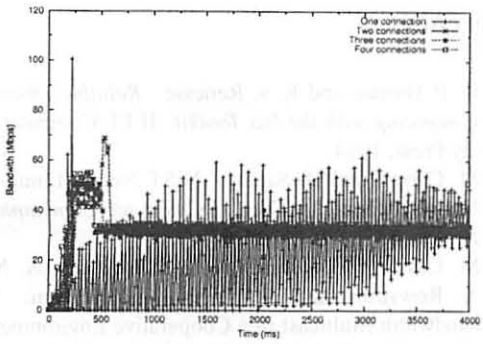

Figure 4. Stability of bandwidth.

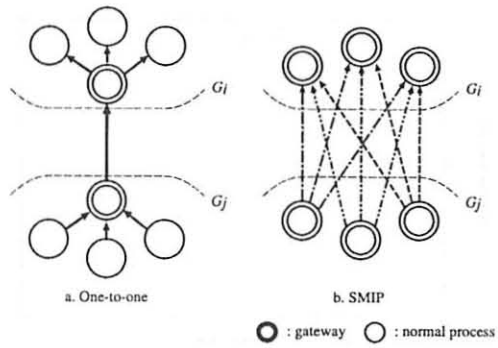

Figure 5. Data transfer arrangement.

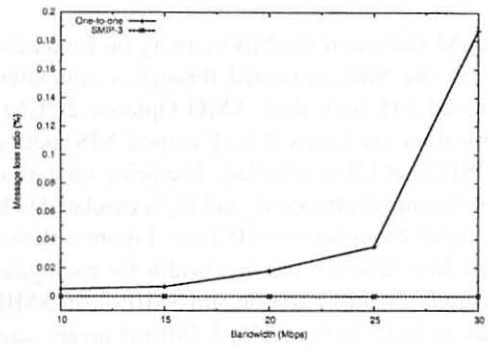

Figure 6. Message loss ratio.

$ssThresh := cwnd_k/2$;
$cwnd_k := cwnd_k * 0.9$;
else $cwnd_k := cwnd_k + 1$;
else /* message loss */
$ssThresh := cwnd_k/2$;
Congestion avoidance phase is started;

3. Congestion avoidance phase at a channel c_k :
   $wnd_k := min(cwnd_k, rwnd_k, qwnd)$;
   $qwnd := qwnd - wnd_k$;
   The number wnd_k of messages in XQ
   are moved to XQ_k;
   send messages from XQ_k;
   On receipt of an acknowledgment message,
   if no message is lost,
   $cwnd_k := 1/cwnd_k$;
   $ssThresh := max(ssThresh, cwnd_k/2)$;
   else /* message loss */
   $cwnd_k := ssThresh$;
   Congestion avoidance phase is restarted;
4. Packet loss is detected by timeout :
   $ssThresh := 0$;
   Slow start is started;

4. Evaluation

We evaluate SMIP in terms of the stability of bandwidth, the message loss ratio, and the delay time compared with the traditional one-channel transmission protocol like TCP. In the traditional one-to-one communication approach, protocols at a lower layer than the transport layer are used to support QoS required by applications. In our striping multi-channel approach, QoS is supported on the end-to-end basis with QoS control at the 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.

A source gateway process in a subgroup G_i is realized in a computer Dell Precision 530 with dual Intel Pentium Xeon 1.8Ghz and 1.5GB memory on Linux 2.6.10. Four destination gateway processes in a subgroup G_j are realized in HP Proliant BL10e blade server with Intel PentiumM 1Ghz and 512MB memory on Linux 2.4.26. These gateways 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 [2] is installed. The delay time between source and destination gateway processes is emulated to be 40 milliseconds by using the NISTnet.

In the evaluation, the source gateway sends multimedia like DV data with 30Mbps. The NewReno algorithm [4] of TCP is used for transmitting messages in each channel. The data transmission procedure of TCP is emulated over UDP/IP. Figure 4 shows the striping multi-channel way supports more stable transmission than the one-channel way. In 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. Even if QoS is degraded in a channel, messages which cannot be transmitted in the channel can be transmitted through the other channels in SMIP.

Next, we measure the message loss ratio. In the traditional way, one gateway in G_i communicates with one gateway in G_j [Figure 5a]. The inter-subgroup communication from l gateways to l gateways is denoted by SMIP-l. Figure 5b shows SMIP-3. Each pair of gateways are interconnected in the 100Mbps Fast Ethernet. Each of normal processes and gateways is realized in an HP Proliant BL10e blade server with Intel

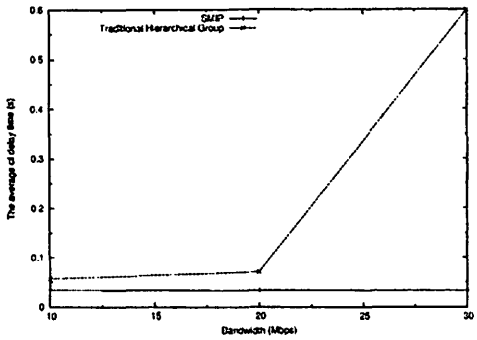

Figure 7. Delay time.

PentiumM 1Ghz and 512MB memory on Linux 2.4.26. Gateways 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 [2] is installed. The delay time of a pair of the subgroups between G_i and G_j is emulated to be 40 milliseconds by using the NISTnet. Figure 6 shows the message loss ratio for the bandwidth for each gateway for the traditional one-to-one and SMIP-3. In SMIP, no message is lost. In Figure 6, k [Mbps] means each of three gateways sends messages at rate $k/3$ [Mbps]. On the other hand, the message loss ratio is increased as the transmission bandwidth of each gateway is increased.

Finally, we measure the delay time. We take a same environment as discussed in message loss ratio. Figure 7 shows the delay time for the bandwidth for each gateway for the traditional one-to-one protocol and SMIP-3. On the other hand, the delay time is increased as the transmission ratio of each gateway is increased.

5. Concluding Remarks

We discussed the hierarchical group. In order to improve the reliability and throughput of the inter-subgroup communication, a pair of parent and child subgroups are interconnected through multiple channels between multiple gateway processes in the subgroup. We discussed the congestion control algorithm for striping 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 SMIP can support the higher stability of the bandwidth and the smaller message loss ratio compared with the traditional protocol.

Acknowledgment

This research is partially supported by Research Institute for Science and Technology [Q05J-04] and Frontier Research and Development Center [16-J-6], Tokyo Denki University.

References

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

[2] M. Carson and D. Santay. NIST Net: a Linux-based Network Emulation Tool. *Computer Communication Review*, 33(3):111–126, 2003.

[3] 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.

[4] S. Floyd, ICSI, T. Henderson, Boeing, A. Gurtov, and TeliaSonera. The NewReno Modification to TCP's Fast Recovery Algorithm. *Request for Comments 3782*, 2004.

[5] V. Jacobson. Congestion Avoidance and Control. In *Proc. of the ACM Symposium on Communications Architectures and Protocols (SIGCOMM '88)*, pages 314–329, 1988.

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

[7] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Trans. on Parallel and Distributed Systems*, 14(3):248–258, 2003.

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

[9] F. Mattern. Virtual Time and Global States of Distributed Systems. *Proc. of the International Workshop on Parallel and Distributed Algorithms*, pages 215–226, 1989.

[10] 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.

[11] 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.

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

[13] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. *Request for Comments 2001*, 1997.

[14] 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.