マルチメディア通信と分散処理 82－5
（1997. 4. 24）

Design and Evaluation of Wide-area Group Communication Protocols - International Experiment

Takayuki Tachikawa, Hiroaki Higaki, Makoto Takizawa(Tokyo Denki Univ.),
Mario Gerla(UCLA), Ming T. (Mike) Liu(Ohio State Univ.),
S. Misbah Deen (Keele Univ.), and Norio Siratori(Tohoku Univ.)
e-mail {tachi, hig, taki}@takilab.k.dendai.ac.jp

In distributed applications, a group of multiple processes is required to be cooperated by exchanging multimedia data. In addition, world-wide distributed applications are being realized by using the Internet. The traditional group communication protocols assume that every pair of processes support almost the same and fixed delay time and reliability level. In world-wide multimedia applications, the assumption does not hold. We discuss high-speed protocols which can change the ways for distributing messages to multiple destinations and retransmitting messages to processes losing the messages in the wide-area group in the change of the delay and reliability. We present the evaluation of the protocols in the world-wide environment.

広域グループ通信プロトコルの設計および評価 - 国際実験

立川 敬行　桧垣 博章　滝沢 誠 (東京電機大)　Mario Gerla (UCLA)
Ming T. (Mike) Liu (OSU)　S. Misbah Deen (Keel University)　白鳥 則郎 (東北大)

複数プロセスが協調動作を行うためのグループ通信では，メッセージを一定順序で配送し，全宛先での受信を保障する必要がある．これまでのグループ通信は，伝搬遅延時間の差が小さいLAN環境が考えられてきた．本論文では，インターネットで広域に分散しているプロセス間のグループ通信を考える．このような広域環境では，グループ内のプロセス間でのメッセージの伝搬遅延時間とメッセージの紛失率が地理的な環境と時間により異なる．本論文では，プロセス間の伝搬遅延時間とメッセージ紛失率がプロセス間と時間により異なるもとで，効率的な再送方式，送信方式を動的に選択するプロトコルを示し，その評価を行う．

1 Introduction

In distributed applications like teleconferences, a group of multiple processes is first established and then the processes in the group are cooperated. Group communication protocols support a group of the processes with the reliable and ordered delivery of messages to multiple destinations. ISIS(CBCAST) [1], and others [7, 9] support the causally ordered delivery. ISIS(ABCAST) [1], and others [2, 8] support the totally ordered delivery.

The group communication protocols discussed so far assume that every pair of processes support almost the same delay time and reliability. That is, only processes in a local area are cooperated. High-speed group communication among multiple processes distributed in a wide area is required to realize the world-wide multimedia applications. Here, let us consider a world-wide teleconference among five processes K, U, S, T, and H at Keele in UK, UCLA and Ohio State Univ. in the USA, and Tohoku Univ. and Dendai, Hatoyama in Japan, respectively. In the Internet, it takes about 60 msec to propagate a message in Japan while taking about 240 msec between Tokyo and Europe. Over than 10% of the messages are lost between Japan and Europe while less than 1% are lost in Japan. Thus, it is essential to consider a *wide-area* group of processes where the delay times and reliability levels between the processes are significantly different [3–5]. In the wide-area group, the time for delivering messages to the destinations is dominated by the longest delay between the processes. For example, if T sends m to H and K, T has to wait for the response from K after having received the response from H. Next, suppose that K sends a message m to H and T, respectively. If T loses m, T requires the sender K to resend m. The delay time between T and K is about four times longer than T and H. If the destination H resends m, the delay time for delivering m can be reduced.

Suppose that T sends m to H, U, and K. On receipt of m, the destination processes send the receipt confirmation messages to T. Here, let us consider a way that K sends the confirmation to U instead of directly sending to T and then U sends the confirmation back to T. Even if U loses m, the delay time can be reduced if K retransmits m to U as presented before. A wide-area group G can be decomposed into disjoint subgroups G_1, ..., G_{sg} ($sg \geq 2$) [3,11]. Holbrook, et. al. [4] presents a way where each subgroup has a message log to retransmit messages. The protocols [3–5, 11] are discussed to reduce the number of messages in

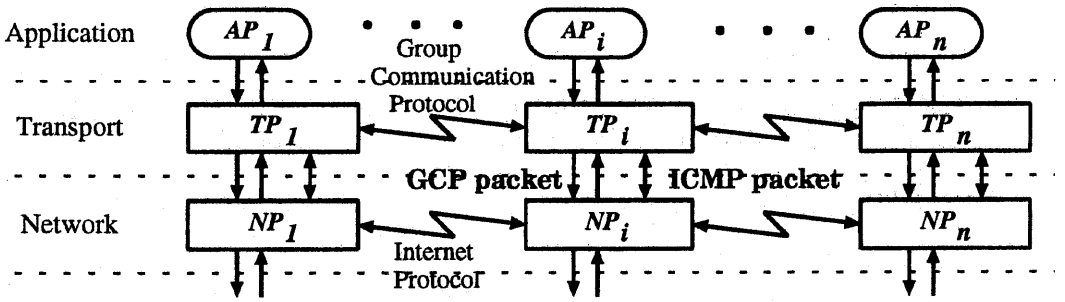

Figure 1: Distributed system

large-scale groups.

Jones, et. al. [5] discuss the saturation protocol where the sender sends multiple replicas of a message m to the destinations. In this paper, we discuss the *destination replication* where the destinations forward m to the other destinations on receipt of m.

In sections 2 and 3, we present a system model and the measurement of the delay time and message loss ratio in the network. In section 4, we discuss ways to reliably and efficiently deliver messages in the wide-area group. In section 5, we present protocols in the wide-area group. In section 6, we present the evaluation of the protocols.

2 System Model

A distributed system is composed of *application*, *transport*, and *network* layers as shown in Figure 1. A group of n (≥ 2) application processes AP_1, ..., AP_n are communicated by using the underlying group communication service supported by transport processes TP_1, ..., TP_n. A group G of the transport processes ($G = \{ TP_1, ..., TP_n \}$) is considered to support each pair of processes TP_i and TP_j with a logical channel. Data units transmitted at the transport layer are *packets*. TP_i sends a packet to TP_j by the channel. The network layer provides the IP service for the transport layer. That is, IP packets may be lost, out of order, and duplicated.

The cooperation of the processes at the transport layer is coordinated by *group communication* (GC) and *group management* (GM) protocols. The GC protocol first establishes a group G and then reliably and causally [1] delivers packets to the processes in G. The GM protocol is used for monitoring and managing the membership of G. AP_i requests TP_i to send an application stream s. TP_i decomposes s into packets, and sends them to the destinations in G. The destination TP_j assembles the packets into stream s_j, and delivers s_j to AP_j. Packets decomposed from the stream are *messages*. Let $dest(m)$ be a set of destination processes of a message m in G.

A transport process TP_i has to know the delay time δ_{ij} and message loss ratio ε_{ij} with each TP_j in G. In the GM protocol, TP_i requests periodically the network layer to transmit two kinds of ICMP packets to all the processes in G: "Timestamp" and "Timestamp Reply". TP_i can know when "Timestamp" sent by TP_i is received by TP_j, and when "Timestamp Reply" received by TP_i is sent by TP_j, i.e. round trip time. TP_i calculates δ_{ij} by using the time information. In addition, the GM protocol monitors the ratio ε_{ij} of packets lost between each pair of TP_i and TP_j. Here, $\overline{\delta}_{ij}$ and $\overline{\varepsilon}_{ij}$ show the averages of δ_{ij} and ε_{ij}, respectively. TP_j is *nearer* to TP_i than TP_k if $\overline{\delta}_{ij} < \overline{\delta}_{ik}$. Here, we assume that $\overline{\delta}_{ij} = \overline{\delta}_{ji}$ and $\overline{\varepsilon}_{ij} = \overline{\varepsilon}_{ji}$ for every pair of TP_i and TP_j.

3 Network Measurement

In the world-wide environment, we measure the delay time δ_{ij} and message loss ratio ε_{ij} for processes TP_i and TP_j. First, the delay times among the processes H at Hatoyama, S at Sendai, U at UCLA, and K at Keele are measured. The process H sends 5,000 'PING' packets to S, U, and K. Each destination process monitors the delay time of each packet received. Here, $R_{ij}(t)$ shows the ratio of the packets which it takes t time units to arrive at the destination TP_j from TP_i to the total number of packets transmitted, i.e. 5,000. In Figure 2, Sendai, UCLA, and Keele show $R_{HS}(t)$, $R_{HU}(t)$, and $R_{HK}(t)$, respectively. The longer the distance is, the more fluctuated the receipt ratio is. For example, 80% of packets arrive at S from H in 50 msec after the fastest packet arrives, i.e. $\int_{30}^{80} R_{HS}(t)dt = 0.8$. Here, it takes about 30 msec for the fastest packet to arrive at S. On the other hand, 80% of the packets arrive at U and K in 85 and 137 milliseconds after the fastest packet arrives there, respectively. If a process TP_i does not receive the confirmation from TP_j in some time units after sending a message m, TP_i considers that TP_j loses m. If the timeout period is given 2 × 50 msec, about 20% of packets are considered to be lost between H and S. In order to receive more than 80% of packets between H and U and between H and K, the timeout period has to be larger than 2 × 85 and 2 × 137 msec, respectively.

In addition, the message loss ratios for S, U, and K are measured as shown in Table 1. Only 1% of the messages are lost for S, i.e. $\int_{30}^{\infty} R_{HS}(t)dt = 0.99$ but 8.3% and 11.7% of the packets are lost for U and K, respectively. It takes averagely 241 msec to deliver a message

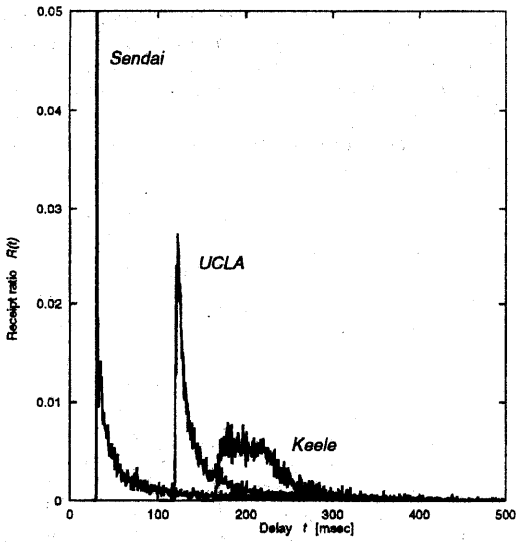

Figure 2: Message receipt ratio v.s. delay.

from H to K while even the fastest packet takes 164.5 msec. Table 1 shows that the longer the distance is, the more messages are lost.

Table 1: Delay[msec] & lost[%].

host	min	avrg	max	lost
Sendai	30.437	60.427	756.263	0.9
UCLA	119.506	157.171	532.433	8.3
Keele	164.497	241.370	2733.565	11.7

Then, we measure how the delay time and message loss ratio are changed from hour to hour in one day. Figure 3 and Figure 4 show the delay time and message loss ratio for each hour. As shown in the figures, they are hour-variant. For example, less than 5% of the packets are lost from 7 o'clock to 17 o'clock while more packets are lost before 7 o'clock and after 17 o'clock between H and K. Figure 3 and Figure 4 show that the longer the distance is, i.e. more routers are hopped, the bigger the variances of the delay time and message loss ratio are. Hence, each process is required to change a transmission and retransmission way according to the changes of delay time and message loss ratio. For example, the sender retransmission is adopted if the message loss ratio is smaller. If the message loss ratio and delay get larger the destination retransmission is adopted.

4 Reliable Delivery

4.1 Transmission and confirmation

In the group communication, each process TP_i sends a message m sent to multiple destination processes in a group $G = \{TP_1, ..., TP_n\}$. Here, let s be the number of the destinations of m, i.e. $s = |dest(m)|$. There are two points to be discussed to realize the reliable receipt of m in G:

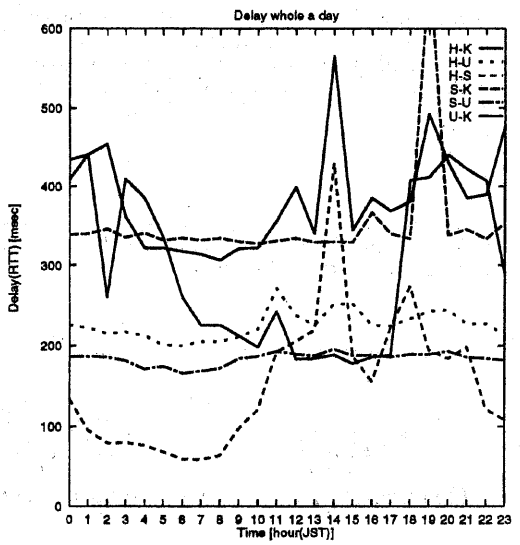

Figure 3: Delay (RTT) a day.

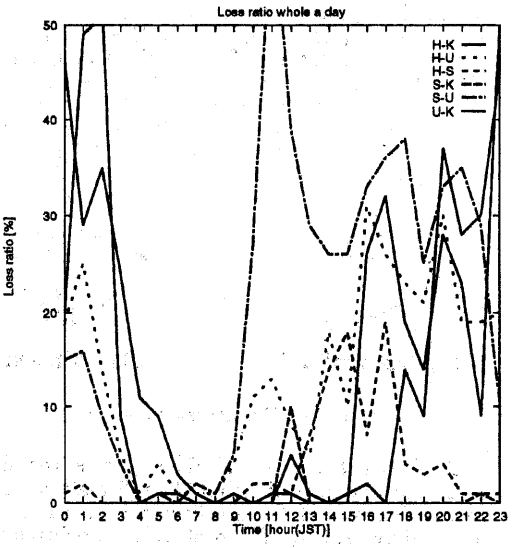

Figure 4: Loss-ratio a day.

(1) how to deliver m to the destinations, and
(2) how to deliver the receipt confirmation of m to the sender TP_i and the destinations.

There are *direct* and *hierarchical* ways [Figure 5] for (1). In the direct multicast, TP_i sends m directly to all the destinations. In the hierarchical multicast, TP_i sends m to a subset of the destinations. On receipt of m from TP_i, TP_j forwards m to other destinations. The propagation tree based routing algorithms [2] are discussed so far. Another example is to decompose G into disjoint subgroups G_1, ..., G_{sg} ($sg \geq 2$) [11]. Each G_i has one coordinator process. TP_i sends m to the coordinator and the coordinator forwards m to the destinations in the subgroups.

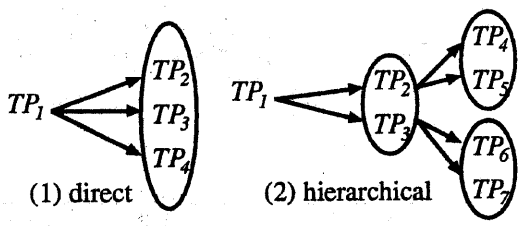

Figure 5: Distribution ways

There are two ways to deliver the confirmation [Figure 6]. In the *decentralized* way [1], the destinations send back the receipt confirmation of m to the sender TP_i. On receipt of all the confirmations, TP_i informs all the destinations of the reliable receipt of m. Totally $3s$ messages are transmitted and it takes three rounds. In the *distributed* way [8, 9], every destination TP_j sends the receipt confirmation of m to all the destinations and TP_i on receipt of m. If each TP_j receives the confirmations from all the destinations, TP_j reliably receives m. Here, $O(s^2)$ messages are transmitted and it takes two rounds. Tachikawa and Takizawa [9] show that $O(s)$ messages are transmitted in G by adopting the *piggy back* and the *deferred confirmation*.

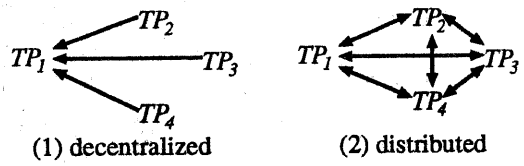

Figure 6: Confirmation ways

A *distributed* protocol [7] supports the direct multicast and distributed confirmation. Direct multicast and decentralized confirmation are adopted by ISIS [1]. The protocols are referred to as *decentralized*.

4.2 Retransmission

In the underlying network, messages are lost due to buffer overruns, unexpected delay, and congestion. Hence, the processes have to recover from the message loss. Let us consider a group $R = \{ H, U, O, K \}$. Suppose that H sends a message m to U, O, and K, but O loses m. In the traditional protocols, H retransmits m to O and it takes $2\overline{\delta}_{HO}$. In another way, U forwards m to O on behalf of H. Here, it takes $2\overline{\delta}_{UO}$. $\overline{\delta}_{HO} > \overline{\delta}_{UO}$. Thus, m can be retransmitted to TP_j by one destination of m, say TP_k whose $\overline{\delta}_k \cdot (1 + \overline{\varepsilon}_{kj}) / (1 - \overline{\varepsilon}_{kj})$ is the minimum if TP_j loses m.

(1) Sender retransmission: The sender TP_i retransmits m to TP_j [Figure 7(1)].
(2) Destination retransmission: Some destination TP_k forwards m to TP_j [Figure 7(2)].

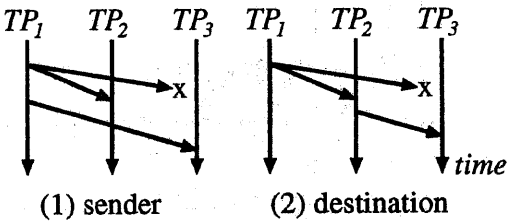

Figure 7: Retransmission

4.3 Replication

If H sends multiple replicas of m to U, U can more surely receive one replica. Another way is that a destination TP_k forwards m to another destination TP_j while TP_i sends m to TP_j. TP_j receives two replicas of m from TP_i and TP_k. For example, U sends m to O on receipt of m while H directly sends m to O. O can receive m from U even if m sent by H is lost.

(1) Sender replication: TP_i sends multiple replicas of m to TP_j.
(2) Destination replication: TP_k receiving m sends m to TP_j.

The sender replication is similar to the saturation protocol [5]. The protocols with the replication are named *replicated* protocols.

There are two kinds of the destination replication. First, one destination TP_k sends one, possibly multiple replicas of m to TP_j on receipt of m [Figure 8(1)]. Secondly, multiple destination processes, say TP_k and TP_l send the replicas to TP_j [Figure 8(2)]. TP_j receives multiple replicas from TP_i and TP_k.

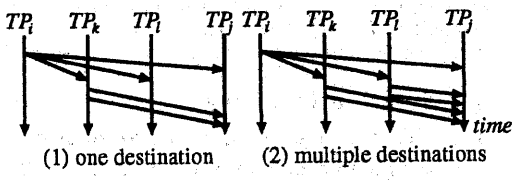

Figure 8: Destination replication

In the *continuous* replication, TP_i sends the replicas of m continuously to TP_j. In another *dis-*

crete replication, TP_i sends the succeeding replicas of m some time units after TP_i sends each replica of m. If the message loss occurs in a burst manner, the discrete replication has to be adopted.

Suppose that TP_i sends h_{ij} replicas of m to TP_j in the sender replication. If TP_j receives no replica from TP_i, TP_i sends h_{ij} replicas of m to TP_j again. The expected time T_{ij} for TP_j to receive at least one replica of m from TP_i is $\delta_{ij} \cdot (1 + \overline{\varepsilon}_{ij}^{h_{ij}}) / (1 - \overline{\varepsilon}_{ij}^{h_{ij}})$. The probability Π_{ij} that TP_j receives at least one replica of m is $1 - \overline{\varepsilon}_{ij}^{h_{ij}}$. The cost C_{ij} for TP_i to send m to TP_j is defined to be $h_{ij} \cdot \delta_{ij}$. In the destination replication, TP_i sends h_{ik} replicas of m by the sender replication. On receipt of at least one replica of m, TP_k sends h_{kj} replicas of m to TP_j. The expected time T_{ikj} for TP_j to receive at least one replica of m forwarded by TP_k is $\overline{\delta}_{ik} \cdot (1 + \overline{\varepsilon}_{ik}^{h_{ik}}) / (1 - \overline{\varepsilon}_{ik}^{h_{ik}}) + \overline{\delta}_{kj} \cdot (1 + \overline{\varepsilon}_{kj}^{h_{kj}}) / (1 - \overline{\varepsilon}_{kj}^{h_{kj}})$. The cost C_{ikj} is $h_{ik} \cdot \delta_{ik} + h_{kj} \cdot \delta_{kj}$. The probability Π_{ikj} that TP_j receives at least one replica of m is $(1 - \overline{\varepsilon}_{ij}^{h_{ij}}) + \overline{\varepsilon}_{ij}^{h_{ij}} \cdot (1 - (1 - \overline{\varepsilon}_{ik}^{h_{ik}}) \cdot (1 - \overline{\varepsilon}_{kj}^{h_{kj}}))$. The destination replication is more efficient than the sender replication if $T_{ij} \geq T_{ikj}$, $\Pi_{ij} \leq \Pi_{ikj}$, and $C_{ij} \geq C_{ikj}$. In the time critical applications, the destination replication can be adopted if $T_{ij} \geq T_{ikj}$ even if $C_{ij} \leq C_{ikj}$. TP_k is selected to forward m to TP_j if T_{ikj} is the minimum among the destinations of m. Here, no sender replication is adopted if $h_{ij} = 1$ and $h_{ik} = 1$.

Let ε be a maximum allowable loss ratio. If $\overline{\varepsilon}_{ij} \leq \varepsilon$, TP_i does not need to send multiple replicas of m to TP_j. If $\overline{\varepsilon}_{ij} > \varepsilon$, TP_i has to send h_{ij} (> 1) replicas of m whether or not TP_i retransmits m. It is required that $\overline{\varepsilon}_{ij}^{h_{ij}} \leq \varepsilon$. Hence, $h_{ij} \geq log\varepsilon \ / \ log\overline{\varepsilon}_{ij}$ if $\overline{\varepsilon}_{ij} \leq \varepsilon$, otherwise $h_{ij} = 1$. For example, $h_{ij} = 3$ if $\varepsilon = 0.01$ and $\overline{\varepsilon}_{ij} = 0.2$. We can consider a way that TP_i sends m to TP_l, TP_l sends m to TP_k, and finally TP_k sends m to TP_j. However we assume that there is only one process TP_k forwarding a replica of m to TP_j.

5 Protocols

We present a protocol for transmitting messages in the group $G = \{ TP_1, ..., TP_n \}$. In the protocol, the messages have to be causally ordered and the messages lost have to be detected. In order to causally order the messages, the vector clock [6] is adopted. Since the gap between messages cannot be detected by the vector clock, the protocol detects the message loss by the sequence numbers of the messages. Each TP_i manipulates the variables. VC_1, ..., VC_n showing the vector clock to causally order the messages received. TP_i manipulates another kind of variables $DCV = \langle DVC_1, ..., DVC_n \rangle$ denoting the vector clocks of the message which are most recently delivered. A variable T_i denotes a current local time in TP_i

Each message m is sent to the destination processes, not necessarily all the processes in the group G. m is given a vector $SEQ = \langle SEQ_1, ..., SEQ_n \rangle$ of sequence numbers. If m is destined to TP_j, SEQ_j is incremented by one. Otherwise, SEQ_j is not changed. The variable SEQ_j denotes a sequence number of a message for TP_j.

TP_i manipulates variables REQ_1, ..., REQ_n to receive messages. REQ_j shows a sequence number of a message which TP_i expects to receive next from TP_j. On receipt of a message m from TP_j, if $m.SEQ_i = REQ_j$, TP_i finds that there is no loss of a message sent by TP_j. The messages received are stored in the buffer $RBUF$. The messages in $RBUF$ are causally ordered by using the vector clocks. The messages sent are also stored in the buffer $SBUF$ in the sending order.

Each message m is composed of the following fields:

- $m.SP$ = process TP_i sending m.
- $m.DP$ = collection of destination processes of m.
- $m.SEQ = \langle m.SEQ_1, ..., m.SEQ_n \rangle$ = sequence number of m.
- $m.DT$ = data.

Here, suppose that TP_i sends a data D to the processes TP_{i1}, ..., TP_{il_i} in G. TP_i makes a following message m.

$m.SP = TP_i$; $m.DP = [TP_{i1}, ..., TP_{il_i}]$;
$SEQ_j = SEQ_j + 1$ for every $TP_j \in m.DP$;
$m.SEQ = SEQ$;
$m.DT = D$;

On receipt of message m from TP_j, TP_i checks the sequence number. If message lost is detected, TP_i sends NACK to the nearest process. Otherwise, TP_i delivers the message and sends ACK.

```
if (receive(m) != 0) {
  if (m.SEQ[i] < REQ[j]) {
    send(NACK);
  }
else
  REQ[j] = m.SEQ[i];
  deliver(m);
  send(ACK);
}
```

6 Evaluation of Protocols

6.1 Reliable receipt

The prototypes of the protocol has been implemented to be a group G of five UNIX processes in SPARC workstations, i.e. two (*ktus0*, *ktus1*) in Hatoyama, one (*swan*) in Sendai, Japan, one (*sunshine*) in UCLA, U.S. and one (*nina*) in Keele, UK. We consider two cases: (1) there is no message loss and (2) *ktus0* loses a message m. We measure the delay time where processes send messages of 1024 bytes to all processes in the group. In the decentralized protocol (D), the sender process *nina* retransmits m. In our protocol (M), *ktus1* nearest to *ktus0* forwards m to *ktus0*.

The following events occur in the process:

send: m is sent by the original sender process.

receive: m is received by the destination process.

deliver: m is delivered to an application process.

reliable receive: The sender process knows that m is received by all the destinations.

detect: A destination process detects a loss of m by receiving another process's confirmation of m.

For each event e, let time(e) be time when e occurs. The following kinds of delays are obtained from the times measured:

receipt(R) delay: time(receive) − time(send).

delivery(DL) delay: time(deliver) − time(send).

reliable receipt(RR) delay: time(reliable receive) − time(send).

detect(DT) delay: time(detect) − time(send).

(1) of Table 2 indicates the R, DL, and RR delays for four protocols in the first case. The difference between R and DL shows time for the protocol processing. The difference between R and RR shows time for exchanging the confirmation messages of m. (2) of Table 2 shows the R, DT, DL, and RR delays in the presence of lost messages. The difference between DL and DT shows time for recovering from the message loss by retransmission. For example, *ktus1* forwards m to *ktus0* in the protocol. The difference between DT and DL shows how long it takes to retransmit m.

Following Table 2, the processes can recover from message loss with shorter delay in our protocol than the decentralized one. In addition, the delay time is almost the same as the no-loss case. In the wide-area group, each channel is different in the delay time and message loss ratio. Hence, the messages can be delivered with shorter delay if the messages are sent through channels with the shorter delay and less loss ratio.

Table 2: Delay [msec].

	Protocols	M	D
(1)	receipt(R)	376	376
	delivery(DL)	383	383
	rel. rec. (RR)	724	1128
(2)	detect (DT)	386	762
	receipt (R)	393	1135
	delivery (DL)	394	1139
	rel. rec. (RR)	735	1891

7 Concluding Remarks

It is important and critical to discuss how to realize the high-speed communication of multimedia data among multiple processes in the world-wide area by using the Internet. We have discussed the wide-area group communication including multiple processes interconnected by the Internet. Here, each logical channel between the processes has a different delay time and message loss ratio. In this paper, we have presented ways to reduce the delay time of messages and improve the reliability in the wide-area group, i.e. destination retransmission and replication. We have presented four protocols, i.e. basic, modified, nested group, and decentralized protocols. We have shown that the our protocol implies shorter delay time through the world-wide experiment.

References

[1] Birman, K., Schiper, A., and Stephenson, P., "Lightweight Causal and Atomic Group Multicast," *ACM Trans. Computer Systems*, Vol.9, No.3, 1991, pp.272-314.

[2] Chang, J. M. and Maxemchuk, N. F., "Reliable Broadcast Protocols," *ACM Trans. Computer Systems*, Vol.2, No.3, 1984, pp.251–273.

[3] Hofmann, M., Braun, T., and Carle, G., "Multicast Communication in Large Scale Netwoks," *Proc. of 3rd IEEE Workshop on High Performance Communication Subsystems(HPCS)*, 1995.

[4] Holbrook, H. W., Singhal, S. K., and Cheriton, D. R., "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," *Proc. of ACM SIGCOMM'95*, 1995, pp 328–341.

[5] Jones, M., Sorensen, S., and Wilbur, S., "Protocol Design for Large Group Multicasting: The Message Distribution Protocol," *Computer Communications*, Vol. 14 No. 5, 1991 pp.287–297.

[6] Mattern, F., "Virtual Time and Global States of Distributed Systems," *Parallel and Distributed Algorithms* (Cosnard, M. and Quinton, P. eds.), *North-Holland*, 1989, pp.215–226.

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

[8] Tachikawa, T. and Takizawa, M., "Selective Total Ordering Broadcast Protocol," *Proc. of IEEE ICNP-94*, 1994, pp.212–219.

[9] Tachikawa, T. and Takizawa, M., "Distributed Protocol for Selective Intra-group Communication," *Proc. of IEEE ICNP-95*, 1995, pp.234–241.

[10] Tachikawa, T. and Takizawa, M., "Communication Protocol for Wide-area Group," *Proc. of the International Computer Symposium (ICS'96)*, 1996, pp. NM 158–165.

[11] Takamura, A., Takizawa, M., and Nakamura, A., "Group Communication Protocol for Large Group," *Proc. of IEEE Conf. on Local Computer Networks (LCN-18)*, 1993, pp.310–319.