Group Synchronization Control over Haptic Media in a Networked Real-Time Game with Collaborative Work

Tatsuya Hashimoto and Yutaka Ishibashi
Department of Computer Science and Engineering,
Graduate School of Engineering,
Nagoya Institute of Technology
Nagoya 466–8555, Japan
t_hashi@mcl.elcom.nitech.ac.jp, ishibasi@nitech.ac.jp

ABSTRACT

This paper deals with group (or inter-destination) synchronization control over haptic media in the case where two groups each of which consists of two players play a networked real-time game in which the two players in each group work with each other collaboratively. The group synchronization control adjusts the output timing of haptic media among the players' terminals. We enhance the synchronization maestro scheme, which the authors previously proposed, for the control so that we can handle two reference output timings, to which the players' terminals adjust their output timings. We examine the influence of the determination methods of the reference output timings on the fairness among the players and the efficiency of the work. By experiment, we demonstrate the effectiveness of a method which employs two reference output timings. The method adjusts the output timing of a terminal with the smaller network latency in a group to that in the other group, and it also adjusts the remaining two output timings.

Keywords

Networked real-time game, Collaborative work, Haptic media, Group synchronization control, Fairness, Experiment

1. INTRODUCTION

By using haptic interface devices in networked 3-D virtual environments, we can largely improve the efficiency of collaborative work such as remote surgery simulation and immerse ourselves in playing networked games [1], [2]. However, the network delay and its jitter in a network may degrade the efficiency of the work and the fairness among players because of disturbance of the temporal relations among multiple haptic media streams [2]–[4].

In [3], by making an experiment in which two users manipulate an object collaboratively with haptic interface devices, Hikichi *et al.* investigate the influence of network delay on the manipulation. As a result, they show that a user with the smaller network latency can help (cover) the other user to work. In [4], the authors demonstrate the effectiveness of *group* (or *inter-destination*) *synchronization control* [5], which adjusts the output timing of haptic media among multiple destinations (i.e., terminals), in similar work to that in [3]. They show that by adjusting the output timing of haptic media to the earlier output timing, the group synchronization control improves the efficiency of the work.

On the other hand, in [2], the authors deal with a networked real-time game in which two players play with each other by using haptic interface devices. They illustrate the effectiveness of the group synchronization control under which the output timing of haptic media is adjusted to the later output timing in terms of the fairness between the two players.

In networked real-time games, multiple players often collaborate with each other and fight against other multiple players. When each player has the output timing different from the other players, the fairness among the players are damaged. To solve this problem, we need to carry out group synchronization control. For example, the output timings of all the terminals are adjusted to the latest output timing among the timings under the control. However, since multiple players do collaborative work in the games, the efficiency of the work may be improved by adjusting the output timings to the earliest output timing. This is because a player with an earlier output timing can help other players with later output timings. Thus, we should study group synchronization control in this case. However, to the best of the authors' knowledge, there is no paper which addresses this issue.

This paper deals with group synchronization control over haptic media in the case where two groups each of which consists of two players play a networked real-time game in which the two players in each group do collaborative work in a 3-D virtual space by using haptic interface devices. We also investigate how to adjust the output timing among the players' terminals by experiment. We enhance the group synchronization control so as to achieve fine-grained adjustment of the output timing by handling two reference output timings. Furthermore, we examine the effects of the group synchronization control from the viewpoints of the fairness and the efficiency of the work.

Netgames'06, October 30–31, 2006, Singapore.

The remainder of the paper is organized as follows. Section 2 describes the networked real-time game with collaborative work. Section 3 outlines the group synchronization control, enhance the control, and explain how to adjust the output timing under the enhanced control. Section 4 illustrates the method of the experiment, and experimental results are presented in Section 5. Section 6 concludes the paper.

2. NETWORKED REAL-TIME GAME

In this section, we explain the networked real-time game with collaborative work. We also describe a system model of the game.

2.1 Game with Collaborative Work

As shown in Fig. 1, two groups (referred to as *groups* α and β in this paper) each of which consists of two players play the networked real-time game in a 3-D virtual space (height: 120 mm, width: 160 mm, depth: 70 mm). The two players in each group collaborate with each other and fight against those in the other group by using haptic interface devices. We employ the PHANToM Omni [6] (just called PHANToM here) as a haptic interface device.

Each player manipulates the *cursor* of the PHANToM in the 3-D virtual space. The cursor denotes a point which the player tries to touch with the device. The two players of each group cooperatively lift and move an *object* (a rigid cube with a side of 30 mm and with a mass of 500 g) which is assigned to the group so as to contain the *target* (a sphere with a radius of 12.5 mm) by putting the object between the two cursors of the PHNAToMs. If the object is not pushed from both sides strongly to some extent, it drops on the floor. The gravitational acceleration is assumed to be 2.0 m/s^2. Each object does not collide with the target or the other object. If the distance between the center of the object and that of the target is less than 8 mm, we judge that the object contains the target. When the target is contained by either of the two objects, it disappears and then appears at a randomly-selected position in the space. The two groups compete on the number of eliminated targets with each other.

2.2 System Model

We show a system model of the networked real-time game based on the client-server model in Fig. 2, where we show what kinds of functions the *server*, four *clients*, and *synchronization maestro* have. We will explain the synchronization maestro in Section 3. Since each client has the same functions as the other clients, we show the functions only at client 1 in the figure. We assume that the timers of each client, the server, and the synchronization maestro are globally synchronized with each other [7].

Each client has the PHANToM. Since the client performs haptic simulation by repeating the servo loop [8] at a rate of 1 kHz, it inputs/outputs a stream of haptic media units (MUs), each of which is the information unit for media synchronization, at the rate; that is, an MU is input/output every millisecond. Each MU contains the identification (ID) number of the client, the positional information of the cursor of the PHANToM, and the sequence number of the servo loop, which we use instead of the timestamp of the MU. MUs input at each client are transmitted to the server.

The server carries out causality (i.e., ordinal relation in this paper) control [9] over received MUs as in [10]. The causality control is required to maintain the temporal order of manipulation events. Each haptic MU has a time limit which is equal to the generation time of the MU plus Δ milliseconds (in the experiment, $\Delta = 5$ ms). If the MU is received by the server before the time limit, it is held in the buffer by the time limit; the server arranges MUs in the buffer according to their timestamps (i.e., the sequence numbers). Then, the server calculates the positions of objects every millisecond by using the information about the position of the cursor of the PHANToM included in the MU. Otherwise, the MU is immediately used for the calculation.

Since there are the four clients, the server would always use four MUs for the calculation every millisecond if there were no network delay jitter. However, there exists network delay jitter. Therefore, the server may have more or less than four MUs to be used for the calculation every millisecond. If it judges that either of the objects contains the target, the position of the target is updated. It also transmits the positional information of the objects, the target, and the four cursors as an MU to the four clients.

When each client receives an MU, the client updates the positions of the objects after carrying out media synchronization control, which includes group synchronization control [10], and calculates the reaction force applied to the user. The rendering rate of the haptic media is 1 kHz at the client as described earlier. Also, the rendering rate of the virtual space is 30 Hz.

This paper deals with two cases depending on whether group synchronization control is exerted or not. We employ and enhance the *synchronization maestro scheme* [11] for group synchronization control. In the synchronization maestro scheme, the *Virtual-Time Rendering* (*VTR*) algorithm [12] is used for intra-stream synchronization control [10], [11]; note that we do not need to carry out inter-stream synchronization control [13] in this paper since the server sends a single haptic media stream to each client. The VTR algorithm dynamically changes the buffering time of MUs according to the network delay jitter. When we do not perform the group synchronization control, we use *Skipping* [12] and VTR for intra-stream synchronization control. Skipping outputs only the latest arrived MU every millisecond and skips obsolete MUs.

3. GROUP SYNCHRONIZATION CONTROL

As described earlier, we employ the synchronization maestro scheme to carry out group synchronization control. We here describe the synchronization maestro scheme briefly. Then, we explain how to adjust the output timing among the four clients. We also enhance the synchronization maestro scheme.

3.1 Outline of Synchronization Maestro Scheme

In this scheme, the *ideal target output time* [12] of an MU is defined as the time at which the MU should be output in the case where there is no network delay jitter. The ideal target output time of an MU is set to the generation time of the MU plus δ, where δ denotes the *target delay time* [9], which is defined as the time from the moment an MU is generated until the instant the MU should be output, and $0 \leq \delta \leq \Delta_{\rm al}$. We employ the *maximum allowable delay* $\Delta_{\rm al}$ [12] in order to preserve the interactivity of haptic media

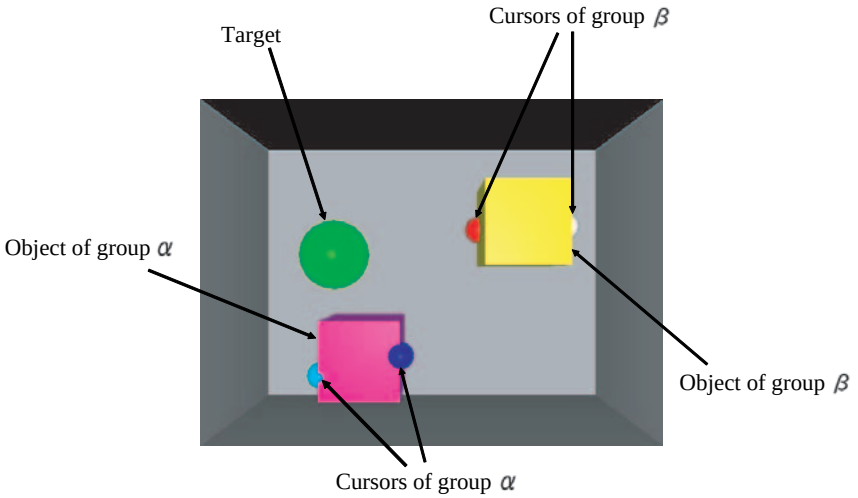

Figure 1: A displayed image of the virtual space.

(in the experiment, the initial value of δ is set to 0 ms, and $\Delta_{\mathrm{al}} = 30$ ms. Other parameters and thresholds are set to the same values as those in [2]).

However, each MU cannot always arrive at each client by the target delay time of the MU owing to network delay jitter. In this case, the value of δ may dynamically be changed by the VTR algorithm [12]. Thus, the *target output time* [12] of an MU, which is defined as the time at which the MU should be output in the case where there exists network delay jitter, may be different from one client to another. To adjust the target output time among the clients, each client transmits the information about the output timing to the synchronization maestro, which plays a role similar to that of an orchestra conductor.

The synchronization maestro determines the *reference output timing* [10] by making a comparison among the output timings received from the clients. Then, the maestro transmits the information about the reference output timing to all the clients.

When each client receives the information about the reference output timing, the client compares its own output timing with the reference output timing. If they are different from each other, the client gradually adjusts its own output timing to the reference output timing. This is because if the client changes the output timing largely at a time, the output quality may be damaged seriously.

For details of the synchronization maestro scheme, the reader is referred to the Appendix.

3.2 Determination Method of Reference Output Timing

We deal with six methods (*methods 1* through *6*) which determines the reference output timing in this paper. Methods 1 through 4 each use a single reference output timing, and methods 5 and 6 each employ two reference output timings.

(Method 1): The latest output timing among the four output timings is selected as the reference one [7].

(Method 2): The earliest output timing among the four output timings is selected as the reference one [7].

(Method 3): We choose the later output timing as the reference one between the earlier one in a group and that in the other group.

(Method 4): We choose the earlier output timing as the reference one between the later one in a group and that in the other group.

(Method 5): We choose the later output timing as one of the reference output timings between the output timing of a client with the smaller network latency in a group and that in the other group. We also choose the later output timing as the other reference one between the remaining two output timings.

(Method 6): We choose the later output timing as a reference one between the output timing of a client with the smaller network latency in a group and that with the larger network latency in the other group. We also choose the later output timing as the other reference one between the remaining two output timings.

Methods 5 and 6 can achieve finer-grained adjustment of the output timings than methods 1 through 4 by employing two reference output timings.

There are other determination methods of the reference output timings. For example, the averages of the output timings are chosen as the reference ones. Handling the methods is for further study.

3.3 Enhanced Scheme

The synchronization maestro scheme in the Appendix employs a single reference output timing. However, since methods 5 and 6 uses two reference output timings, we enhance the scheme in this paper.

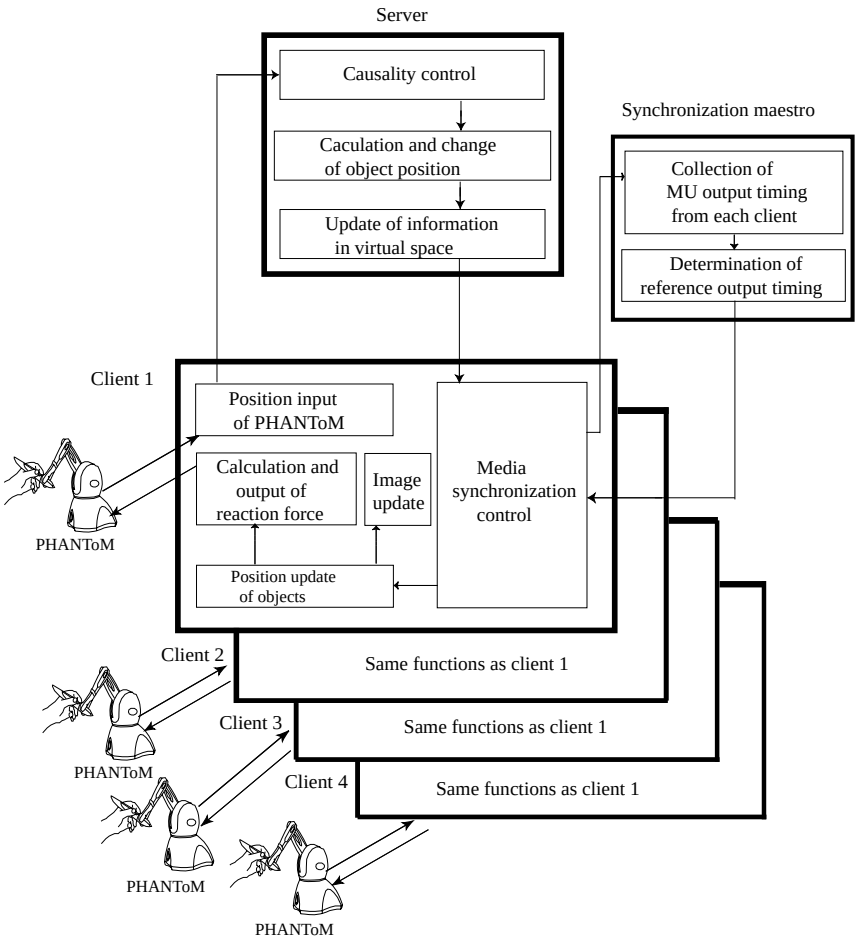

Figure 2: A system model.

In the enhanced scheme, each client sends the synchronization maestro the information about the *average network delay*, which is defined as the average time from the moment an MU is generated at the server until the instant the MU is received at the client for simplicity, as well as the information about the output timing. It should be noted that we here let the average network delay denote the network latency.

In this paper, we obtain the average network delay as follows. Let the network delay of the n-th MU ($n \geq 1$) be denoted by $D(n)$, and the average network delay by $\overline{D}(n)$. The average network delay is calculated by $\overline{D}(n) = P\overline{D}(n-1) + (1-P)D(n)$ [14], where P is a smoothing factor. We set $P = 0.999$ in the experiment.

When the synchronization maestro receives the information about the average network delay and output timing from a client, the maestro compares the average network delay with the other clients' average network delays and determines two pairs of clients (note that each pair consists of a client in group α and that in group β) whose output timings are adjusted based on methods 5 and 6. For example, when the average network delays of clients 1 through 4, which are assumed to belong to groups α, α, β, and β, respectively, are 5 ms, 20 ms, 10 ms, and 30 ms, respectively, clients 1 and 3 make a pair in method 5; clients 2 and 4 make the other pair. Then, the maestro determines the reference output timing for each pair by make a comparison between two output timings of the pair.

Methods 5 and 6 choose the later output timing as the reference output timing as described earlier. This is because we found that methods which choose the earlier output timing as the reference output timing are inferior to methods 5 and 6 by experiment.

4. METHOD OF EXPERIMENT

4.1 Experimental System

As shown in Fig. 3, our experimental system consists of the server (CPU: Pentium4 processor at 2.26 GHz, OS: Windows2000) and clients 1 through 4 (CPU: Pentium4 processor at 2.80 GHz, OS: WindowsXP). The server is connected to the four clients via an Ethernet switching hub (100 BASE-T) and a network simulator (NIST Net [15]). The server has a function of the synchronization maestro for simplicity.

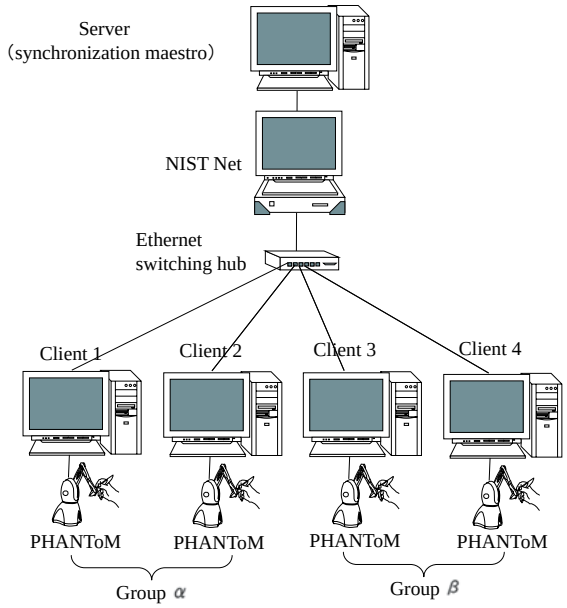

Figure 3: Configuration of the experimental system.

Each MU is transmitted as one packet by UDP. In the experiment, clients 1 and 2 form group α, and clients 3 and 4 group β.

We generate an additional delay for each MU transmitted from the server to each client and for each packet transmitted from the synchronization maestro (i.e., the server) to each client according to the Pareto-normal distribution [15] by using NIST Net[1]. Each MU and each packet are unicast from the server to each client in the experiment.

The additional delay of MU or packet transmitted from the server to client i (i = 1, 2, 3, 4) is referred to as *additional delay* i in this paper. To examine the influences of the difference in the additional delay among the four clients on the fairness and the efficiency of the work, we set the averages of additional delays 1 through 4 as shown in Table 1, where the average of each additional delay is selected from among 0 ms, 10 ms, 20 ms, and 30 ms. We do not handle average additional delays larger than 30 ms since it is difficult to manipulate the object owing to vibrations of the object when the network delay is too large [2]. We also set the standard deviation of each additional delay to 3 ms; however, when the average additional delay is 0 ms, the standard deviation is set to 0 ms.

We handle the following four cases depending on how to select the average additional delay as shown in Table 1.

(Case 1): All the clients have different average additional delays from each other.

(Case 2): Two clients have an identical average additional delay, and the remaining two clients have different average additional delays from the delay.

(Case 3): Only one client has a different average additional delay from the other clients.

(Case 4): All the clients have an identical average additional delay.

Case 1 has three ($=_4C_2/2$) types of measurements (*measurements A*, *B*, and *C*) since we do not need to take account of the difference between the two clients in each group or the difference between groups α and β. In case 2, for simplicity, the remaining two clients are also assumed to have an identical additional delay; *measurement D* has different values in each group, and *measurement E* has the same value in each group. In *measurement F* of case 3, the additional delay of a client is larger than those of the other three clients. In *measurement G*, a client has a smaller average additional delay than the other clients. *Measurement H* has an identical average additional delay at all the clients.

4.2 Performance Measures

As performance measures, we employ the *average total number of eliminated targets* and the *elimination rate of the target* [2]. The average total number of eliminated targets is closely related to the efficiency of the work. The elimination rate of the target is defined as the ratio of the number of targets eliminated by a group to the total number of eliminated targets. This measure denotes the fairness. If the four players have the same skill, we get the elimination rate of 0.5 when there is no difference in the average additional delay among the clients. Before the experiment, the four players practiced many times to eliminate the difference in the skill among them (actually, we will see in Section 5 that the elimination rate is approximately 0.5 when there is no difference

[1]We made the experiment by adding delays to MUs transmitted from each client to the server. As a result, we obtained almost the same results as those in this paper. The results will be presented in another paper.

Table 1: Averages of additional delays 1 through 4 in milliseconds.

		Additional delay 1	Additional delay 2	Additional delay 3	Additional delay 4
Case 1	Measurement A	0	20	10	30
	Measurement B	0	10	20	30
	Measurement C	0	30	10	20
Case 2	Measurement D	10	20	10	20
	Measurement E	10	10	20	20
Case 3	Measurement F	10	30	10	10
	Measurement G	20	10	20	20
Case 4	Measurement H	10	10	10	10

in the average additional delay among the clients). Therefore, the elimination rate of 0.5 implies that the fairness is perfectly maintained between the two groups. A client which each player used was always the same in the experiment. The measurement of the performance was carried out for 30 seconds after the beginning of each experimental run.

5. EXPERIMENTAL RESULTS

We show only the experimental results of measurements A, D, F, and H in this paper. Since measurements B, C, E, and G had almost the same experimental results as measurements A, D, F, and H, respectively, we do not show the results of measurements B, C, E, and G. We show the elimination rate of the target and the average total number of eliminated targets of group α in measurement A in Figs. 4 and 5, respectively. We also show the two performance measures in measurements D, F, and H in Figs. 6 through 11. The four players conducted the experiment 20 times in each measurement in each of Skipping, VTR, and methods 1 through 6. In Figs. 4 through 11, we plot the 95 % confidence intervals.

In Fig. 4, we see that the elimination rates of methods 1, 5, and 6 are approximately 0.5. To clarify the reason, we measured the *average MU output delay*, which is defined as the average time from the moment an MU is generated at the server until the instant the MU is output. As a result, the average MU output delays of group α (i.e, clients 1 and 2) were almost the same as those of group β (clients 3 and 4) in methods 1, 5, and 6. That is, the average MU output delay of client 1 was almost the same as that of the client 3 or 4, and the remaining two clients had almost the same average MU output delay. For example, the average MU output delays of clients 1 through 4 were 10.4 ms, 30.4 ms, 10.3 ms, and 30.4 ms, respectively, in method 5. Thus, in methods 1, 5, and 6, since group α has almost the same output timings as group β, the elimination rates are approximately 0.5. However, the 95 % confidence interval of method 1 is very large. Therefore, methods 5 and 6 can keep the fairness in better condition than the other schemes.

From Fig. 5, we notice that the average total numbers of eliminated targets of Skipping, VTR, and methods 2, 3, and 5 are larger than 15. However, those of methods 1, 4, and 6 are smaller than 15. The average MU output delays of Skipping, VTR, and methods 2, 3, and 5 at clients 1 through 4 were smaller than or almost equal to those of methods 1, 4, and 6. For example, the average MU output delays of clients 1 through 4 were 30.4 ms, 30.3 ms, 30.4 ms, and 30.4 ms, respectively, in method 1, but those in method 2 were 0.4 ms, 20.3 ms, 10.4 ms, and 30.4 ms, respectively. Although methods 5 and 6 can keep the fairness in good condition, the efficiency of the work of method 6 is not high. Therefore, method 5 is superior to the other schemes in terms of the fairness and the efficiency of the work in measurement A.

In Fig. 5, we find that the average total number of eliminated targets of VTR is smaller than that of Skipping. This is because the buffering time of MUs at each client is lengthened and shortened independently of the other clients in VTR. Actually, the average MU output delays of clients 1 through 4 were 0.3 ms, 23.1 ms, 13.0 ms, and 30.1 ms, respectively, in VTR; however, those in Skipping were 0.3 ms, 20.3 ms, 10.4 ms, and 30.4 ms; thus, the average MU output delays of clients 2 and 3 in VTR are somewhat larger than those in Skipping.

Figure 6 reveals that the elimination rates of all the schemes are approximately 0.5 in measurement D. This is because the average additional delays of group α are the same as those of group β; that is, the averages of additional delays 1 and 3 are 10 ms, and those of additional delays 2 and 4 are 20 ms. In fact, the average MU output delays of group α were almost the same as those of group β even without carrying out the group synchronization control.

However, there are differences in the average total number of eliminated targets among the schemes in Fig. 7. The average total numbers of eliminated targets of Skipping, VTR, and methods 2, 3, and 5 are larger than those of methods 1, 4, and 6. The reason can be explained by the difference in the average MU output delay as in the case of measurement A. That is, the average MU output delays at clients 1 through 4 were 10.3 ms to 13.0 ms, 20.3 ms to 23.1 ms, 10.3 ms to 13.1 ms, and 20.3 ms to 23.1 ms, respectively, in Skipping, VTR, and methods 2, 3, and 5, but those in methods 1, 4, and 6 were 20.3 ms or 20.4ms. Thus, we can say that Skipping, VTR, and methods 2, 3, and 5 are effective in measurement D.

In Fig. 8, we observe that the elimination rates of methods 1, 5, and 6 are approximately 0.5 in measurement F. However, method 1 has a large confidence interval. From Fig. 9, we see that the average total numbers of eliminated targets of all the schemes excluding method 1 are larger than 15. The reason is that all the clients had the average MU output delays of around 30.3 ms in method 1; however, some clients had the average MU output delays smaller than 30 ms in the other schemes. Therefore, method 5 and 6 are effective in measurement F.

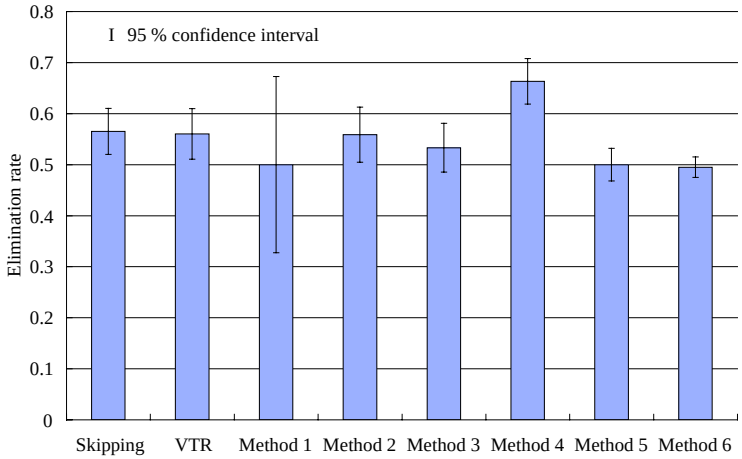

Figure 4: Elimination rate of the target of the group α in measurement A.

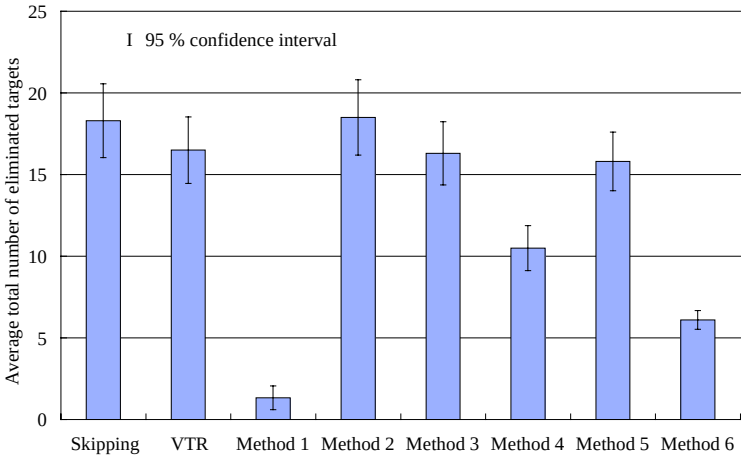

Figure 5: Average total number of eliminated targets in measurement A.

In Fig. 10, the elimination rates of all the schemes are approximately 0.5 in measurement H. Also, the average total numbers of eliminated targets of all the schemes are larger than 15 in Fig. 11. We further notice that the average total number of eliminated targets of VTR is slightly smaller than those of the other schemes. The reason is the same as that in Fig. 5. Therefore, the difference among all the schemes is small in measurement H.

From the above observations, we can say that method 5 is superior to the other schemes in terms of the fairness and the efficiency of the work.

6. CONCLUSIONS

This paper investigated the effects of group synchronization control over haptic media in a networked real-time game with collaborative work by experiment. We enhanced the synchronization maestro scheme for the control so as to handle two reference output timings. As a result, we saw that method 5 can achieve high efficiency of the work while keeping the fairness in good condition; in method 5, we choose the later output timing as a reference one between the output timing of a client with the smaller latency in a group and that in the other group, and we also choose the later one as the other reference one between the remaining two.

As the next step of our research, we plan to carry out subjective assessment of the fairness and the efficiency of the work. We also need to investigate other determination methods of the reference output timing and handle the case of more than four clients. Furthermore, we will deal with networked real-time games different from that handled in this paper.

7. ACKNOWLEDGMENTS

This work was supported by the Grant-In-Aid for Scientific Research of Japan Society for the Promotion of Science un-

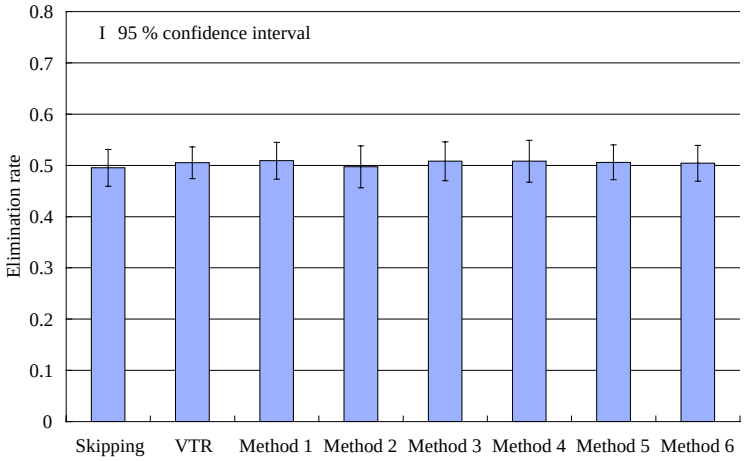

Figure 6: Elimination rate of the target of the group α in measurement D.

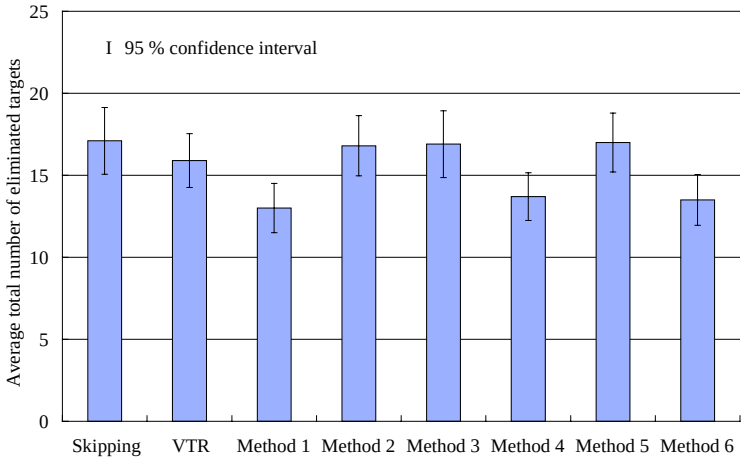

Figure 7: Average total number of eliminated targets in measurement D.

der Grant 16560331.

8. REFERENCES

[1] M. A. Srinivasan and C. Basdogan. Haptics in virtual environments: Taxonomy, research status, and challenges. *Computers and Graphics*, 21(4):393-404, April 1997.

[2] Y. Ishibashi and H. Kaneoka. Fairness among game players in networked haptic environments: Influence of network latency. In *Proc. IEEE ICME'05*, July 2005.

[3] K. Hikichi, H. Morino, I. Arimoto, I. Fukuda, S. Matsumoto, M. Iijima, K. Sezaki, and Y. Yasuda. Architecture of haptics communication system for adaptation to network environments. In *Proc. IEEE ICME'01*, August 2001.

[4] H. Kaneoka and Y. Ishibashi. Effects of group synchronization control over haptic media in collaborative work. In *Proc. the 14th International Conference on Artificial Reality and Telexistence (ICAT'04)*, pages 138-145, November/December 2004.

[5] M. Mauve. Consistency in replicated continuous interactive media. In *Proc. ACM CSCW'00*, pages 181-190, December 2000.

[6] J. K. Salisbury and M. A. Srinivasan. Phantom-based haptic interaction with virtual objects. *IEEE Computer Graphics and Applications*, 17(5):6-10, September/October 1997.

[7] Y. Ishibashi, T. Hasegawa, and S. Tasaka. Group synchronization control for haptic media in networked virtual environments. In *Proc. the 12th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (Haptics'04)*, pages 106-113, March 2004.

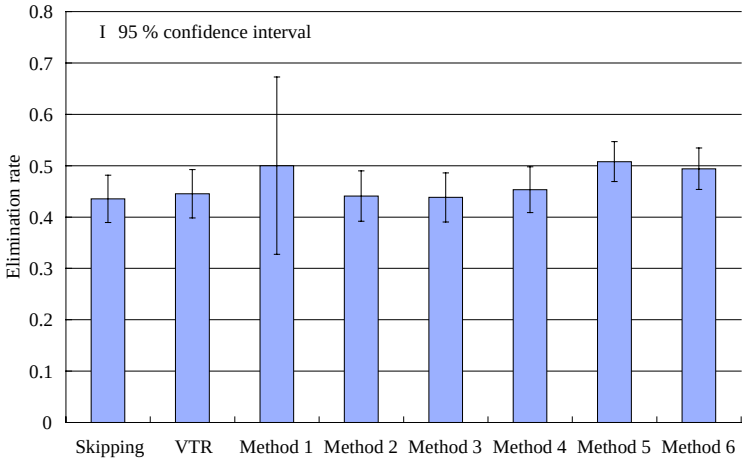

Figure 8: Elimination rate of the target of the group α in measurement F.

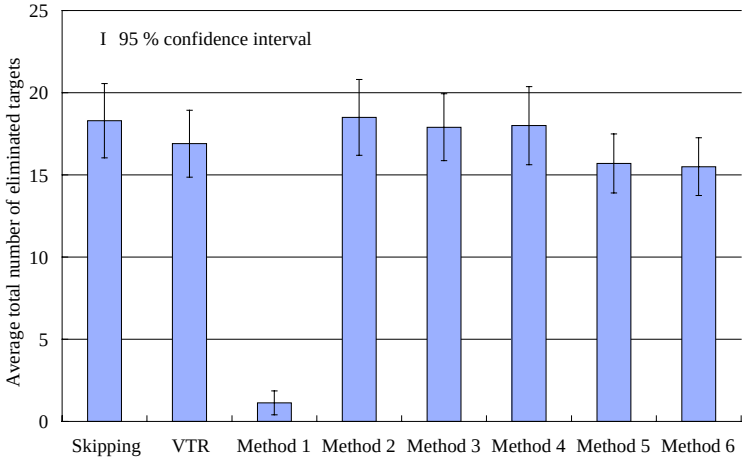

Figure 9: Average total number of eliminated targets in measurement F.

[8] SensAble Technologies, Inc. 3D Touch SDK OpenHaptics Toolkit programmer's guide. Version 1.0, 2004.

[9] Y. Ishibashi, S. Tasaka, and Y. Tachibana. Adaptive causality and media synchronization control for networked multimedia applications. In *Conf. Rec. IEEE ICC'01*, pages 952–958, June 2001.

[10] Y. Ishibashi and H. Kaneoka. Group synchronization for haptic media in a networked real-time game. *IEICE Trans. Commun., Special Section on Multimedia QoS Evaluation and Management Technologies*, E89-B(2):313-319, February 2006.

[11] Y. Ishibashi and S. Tasaka. A group synchronization mechanism for live media in multicast communications. In *Conf. Rec. IEEE GLOBECOM'97*, pages 746-752, November 1997.

[12] Y. Ishibashi, S. Tasaka, and T. Hasegawa. The virtual-time rendering algorithm for haptic media synchronization in networked virtual environments. In *Proc. the 16th International Workshop on Communication Quality & Reliability (CQR'02)*, pages 213-217, May 2002.

[13] Y. Ishibashi, T. Kanbara, and S. Tasaka. Inter-stream synchronization between haptic media and voice in collaborative virtual environments. In *Proc. ACM Multimedia'04*, pages 604-611, October 2004.

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

[15] M. Carson and D. Santay. NIST Net - A Linux-based network emulation tool. *ACM SIGCOMM*, 33(3):111-126, July 2003.

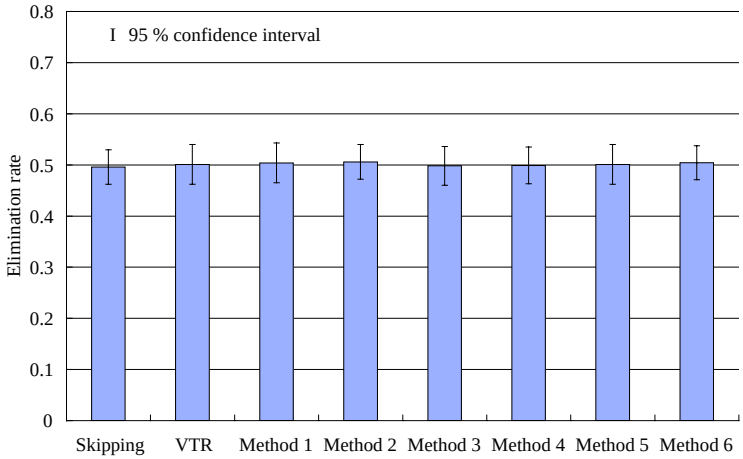

Figure 10: Elimination rate of the target of the group α in measurement H.

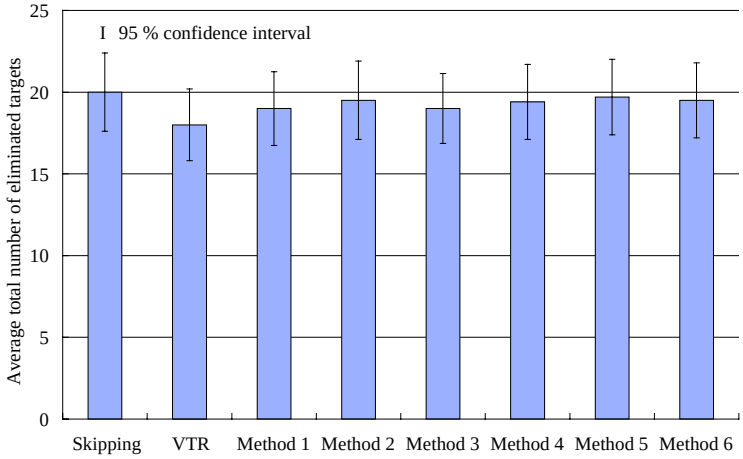

Figure 11: Average total number of eliminated targets in measurement H.

APPENDIX

A. SYNCHRONIZATION MAESTRO SCHEME

In order to explain the synchronization maestro scheme, let us focus on a client. Let x_n denote the ideal target output time (see Subsection 3.1) of the n-th MU ($n = 1,2, \cdots$). Also, let T_n, A_n, and D_n denote the generation time, arrival time, and output time, respectively, of the n-th MU. It should be noted that the values of these variables are integers represented in milliseconds.

The ideal target output time x_n is calculated as follows:

$$x_1 = T_1 + \delta, \tag{1}$$

$$x_n = x_1 + (T_n - T_1) \quad (n \geq 2), \tag{2}$$

where δ denotes the target delay time (see Subsection 3.1).

We cannot always output each MU at its ideal target output time since there exists network delay jitter. Therefore, we further employ the target output time (see Subsection 3.1) t_n of the n-th MU, which is calculated by adding some amount of time (called the *total slide time* [12]) to the ideal target output time.

Let us define the *slide time* [12] of the n-th MU, which is denoted by ΔS_n, as the difference between the *modified target output time* [12] t_n^* and the original target output time t_n. We also define the total slide time S_n as follows:

$$S_0 = 0, \tag{3}$$

$$S_n = S_{n-1} + \Delta S_n \quad (n \geq 1), \tag{4}$$

where $\Delta S_1 = 0$. Then, t_n and t_n^* are expressed by

$$t_1 = x_1, \tag{5}$$

$$t_n = x_n + S_{n-1} \quad (n \geq 2), \tag{6}$$

$$t_n^* = t_n + \Delta S_n \quad (n \geq 1). \tag{7}$$

When the client receives the first MU, it determines the output time D_1 of the MU as follows: $D_1 = \max(t_1, A_1)$. Then, it inquires of the synchronization maestro whether the target output time should be modified or not, by sending the information about the output timing to the maestro. The purpose is to adjust the output timing of the succeeding MUs among all the clients. In this paper, we represent the output timing in terms of the total slide time. Therefore, the client sends a recommended value of the total slide time, which is referred to as the *recommended total slide time* [9], to the maestro.

After the beginning of the output, when the client receives a constant number of consecutive MUs each of which has arrived earlier (or later) than its target output time, it notifies the maestro of the recommended total slide time if it has not transmitted any information to the maestro for those MUs at all. The recommended total slide time is different from the total slide time in that the latter is the accumulation of the slide times, while the former is employed for inquiry about the modification of the target output time in advance. The amount by which the target output time should be modified is called the *recommended slide time* [9]. Let us denote the recommended total slide time and recommended slide time for the n-th MU by s_n and Δs_n, respectively. These times are given by

$$s_1 = \Delta s_1 = D_1 - x_1, \tag{8}$$

$$s_n = S_{n-1} + \Delta s_n \quad (n \geq 2). \tag{9}$$

We will explain how to obtain the value of Δs_n $(n \geq 2)$ later.

In addition, the client notifies the synchronization maestro of the total slide time S_n whenever the target output time is modified [12] at the n-th $(n \geq 2)$ MU (that is, in the case of the *virtual-time expansion*).

When the synchronization maestro receives the total slide time or the recommended total slide time from each client, it determines the reference value S of the total slide time as the reference output timing. Then, the maestro transmits the information about S to all the clients at regular intervals (every 5 seconds in the experiment in Section 4) [9].

The client gradually adjusts its own total slide time to the reference one S when it receives the information about S. Until the client receives the information about S for the first time, it sets the initial value of S to $S_1(= 0)$. For the adjustment, it compares S_{n-1} with S at the n-th $(n \geq 2)$ MU.

First, let us describe the control in the case of $S = S_{n-1}$. Next, we explain the control when $S > S_{n-1}$ and then that when $S < S_{n-1}$.

(a) Case of $S = S_{n-1}$

In this case, if $t_n \geq A_n$ $(n \geq 2)$, the client sets the *scheduled output time* [12] d_n of the n-th MU as follows: $d_n = t_n$ (in this paper, we do not need to perform inter-stream synchronization control as described earlier. Thus, the output time of the n-th MU is set to $D_n = d_n$). Otherwise, it sets $d_n = A_n$. If the MU arrives more than T_{h2} milliseconds later than its target output time (that is, if $A_n - t_n > T_{h2}$), we set the slide time as follows: $\Delta S_n = \min(r_1, T_n + \Delta_{\mathrm{al}} - t_n)$; this means that the virtual-time expansion occurs; note that T_{h2} is a threshold value which we use so as to judge whether the target output time should be delayed or not [12]. In this equation, r_1 $(r_1 \geq 1$ ms) is the maximum value of the slide time in the case where the total slide time is increased under the intra-stream synchronization control, and the smaller value between the two is chosen so that the modified target output time does not exceed the generation time T_n of the MU plus Δ_{al}.

Also, let us assume that when the client receives the n-th MU, it observes that N_{c} $(N_{\mathrm{c}} \geq 1)$ consecutive MUs each have arrived later than their target output times. We further assume that for all the N_{c} MUs the client has sent information about neither the total slide time nor the recommended total slide time to the maestro. Then, the client sets $\Delta s_n = \min(r_2, T_n + \Delta_{\mathrm{al}} - t_n)$ and notifies the maestro of the recommended total slide time s_n, where r_2 $(r_2 \geq 1$ ms) is the maximum value of the recommended slide time for increment of the recommended total slide time. On the other hand, when the client observes that N_{d} $(N_{\mathrm{d}} \geq 1)$ successive MUs each have arrived earlier than their target output times, it sets $\Delta s_n = -\min(r_3, S_{n-1})$ so that the modified target output time does not become less than the ideal target output time, where r_3 $(r_3 \geq 1$ ms) is the maximum absolute value of the recommended slide time for decrement of the recommended total slide time. The client also transmits the information about the value of s_n to the maestro.

(b) Case of $S > S_{n-1}$

When $S > S_{n-1}$, the client sets $\Delta S_n = \min(r_4, S - S_{n-1})$ so as to adjust its total slide time to the reference one (i.e., the virtual-time expansion), where r_4 $(r_4 \geq 1$ ms) is the maximum value of the slide time by which the total slide time is increased under the group synchronization control [9]. However, as described earlier, if we change the total slide time to be adjusted to the reference one whenever the client receives an MU, the output quality of haptic media may deteriorate seriously; it should be noted that the virtual-time expansion brings pausing MUs [12]. Therefore, we adjust the total slide time every N_{e} MUs for each of which the total slide time is larger than the reference one. In this case, if $t_n^* \geq A_n$, the client sets $d_n = t_n^*$; otherwise, it sets $d_n = A_n$.

(c) Case of $S < S_{n-1}$

When $S < S_{n-1}$, the client sets $\Delta S_n = -\min(r_5, S_{n-1} - S)$ every N_{f} MUs for each of which the total slide time is smaller than the reference one as in case (b), where r_5 $(r_5 \geq 1$ ms) is the maximum absolute value of the slide time by which the total slide time is decreased for group synchronization [9]; that is, the *virtual-time contraction* occurs; note that the virtual-time contraction brings skipping MUs [12]. The client determines d_n in the same way as in case (b).

We have a possibility that $d_n \leq D_m$ $(m < n)$ in the case of the virtual-time contraction, where m is the sequence number of the last output MU. In this case, we skip the n-th MU.