PAPER *Special Section on Multimedia QoS Evaluation and Management Technologies*

Group Synchronization for Haptic Media in a Networked Real-Time Game

Yutaka ISHIBASHI[†a)], ***Member and*** **Hiroyuki KANEOKA**[†], ***Student Member***

SUMMARY This paper investigates the effects of group (or inter-destination) synchronization control over haptic media in a networked game where two players move objects competitively by manipulating haptic interface devices. The group synchronization control adjusts the output timing of haptic media among multiple players. By experiment, we demonstrate the effectiveness of the control. We also discuss the fairness between the two players quantitatively.

key words: *networked game, haptic media, group synchronization, experiment*

1. Introduction

In networked 3-D virtual environments which are constructed by computer graphics, we can largely improve the efficiency of collaborative work such as remote surgery simulation and remote design by using haptic interface devices [1], [2]. However, the network delay in the Internet may seriously degrade the efficiency of the work. The network delay jitter also disturbs the temporal relations among multiple haptic media streams. To solve the problem, we need to carry out media synchronization control [3] for haptic media.

Media synchronization control falls into three types: *intra-stream*, *inter-stream*, and *group* (or *inter-destination*) synchronization control. The intra-stream synchronization control is necessary for the preservation of the timing relation between *media units* (*MUs*), each of which is the information unit for media synchronization, in a single media stream. The inter-stream synchronization control is required for keeping the temporal relation among MUs in multiple media streams. Group synchronization control as well as the first two types of control is needed in multicast communications [4], [5]. The purpose of the group synchronization control is to output each MU of media streams simultaneously at different destinations for the fairness among the destinations. This paper focuses on the group synchronization control.

There are a few papers which address the group synchronization issue for haptic media [6]. In [6], the authors enhance the *synchronization maestro scheme* [4], which they previously proposed for voice and video. In the scheme, the *synchronization maestro* gathers the information about the output timing of haptic MUs from each client, and it determines the *reference output timing* and transmits the information about the reference output timing to all the clients. Each client gradually adjusts its output timing to the reference output timing. In [6], by carrying out an experiment in which two clients manipulate an object collaboratively with haptic interface devices (PHANToM DESKTOP [7]), the authors try to demonstrate the effectiveness of the group synchronization control. They deal with two methods depending on which client's output timing is selected as the reference output timing (*Methods 1* and *2*). Method 1 selects the later output timing, and Method 2 the earlier. As a result, they show that Method 2 is better than Method 1; this result is different from the result shown in [5], where Method 1 outperforms Method 2 for voice and video. This is because haptic media have severer requirements for the interactivity than voice and video; the maximum allowable delay is about 30 to 60 ms for haptic media [8]. However, which method is better than the other may depend on the type of work. When we handle different kinds of work from the cooperative work, Method 2 may not be desirable.

This paper deals with a network real-time game (i.e., competitive work) as such work. In this case, the difference in the output timing among multiple players leads to unfairness among them. Therefore, we can guess that Method 1 is superior to Method 2 in the case of two players. However, we cannot find any papers which address the fairness issue for haptic media.

In this paper, we suppose that two players do a networked real-time game in a 3-D virtual space by using haptic interface devices. We also investigate the effects of the group synchronization control in terms of the fairness.

The remainder of the paper is organized as follows. Section 2 describes a system model for haptic media. Section 3 explains the group synchronization control scheme. Section 4 illustrates the experimental system, and the experimental results are presented in Sect. 5. Section 6 concludes the paper.

2. System Model

We suppose a situation in which N ($N \geq 2$) clients move objects competitively by manipulating haptic interface devices in a networked 3-D virtual space (see Fig. 1). Each client has the PHANToM DESKTOP [7] as a haptic interface device. Since the client performs haptic simulation by repeating the servo loop at a rate of 1 kHz [9], it inputs/outputs a stream of

Manuscript received April 18, 2005.
Manuscript revised August 9, 2005.

[†]The authors are with the Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology, Nagoya-shi, 466-8555 Japan.
a) E-mail: ishibasi@nitech.ac.jp

DOI: 10.1093/ietcom/e89–b.2.313

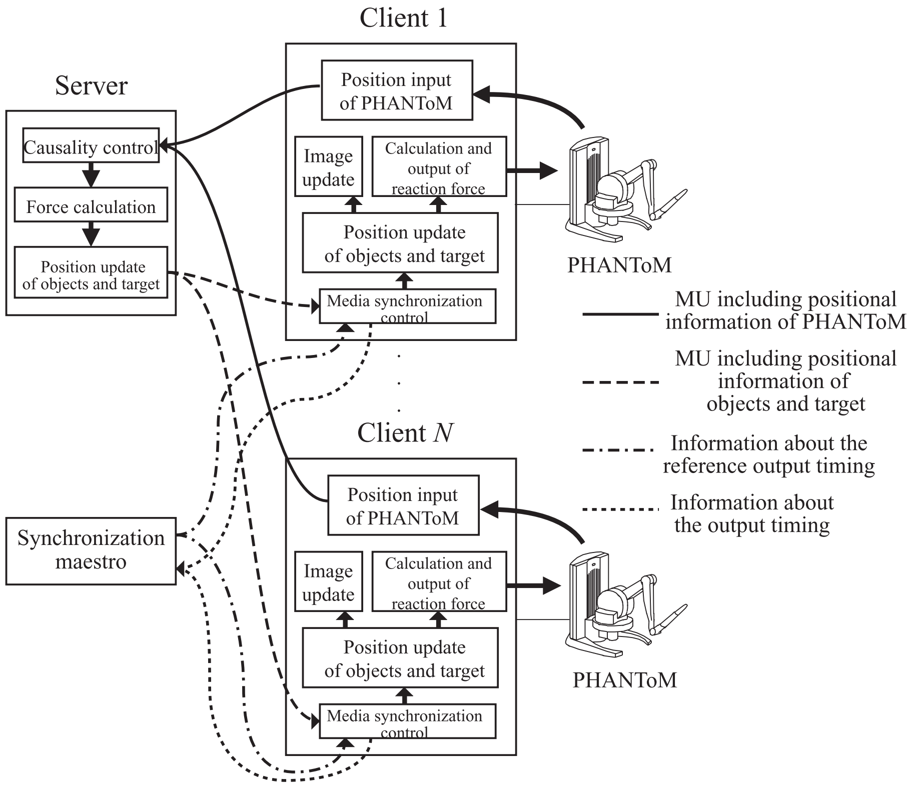

Client 1
Server
Causality control
Force calculation
Position update of objects and target
Position input of PHANToM
Image update
Calculation and output of reaction force
Position update of objects and target
Media synchronization control
PHANToM
Client N
Synchronization maestro
MU including positional information of PHANToM
MU including positional information of objects and target
Information about the reference output timing
Information about the output timing

Fig. 1 A system model.

haptic MUs 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 a single server.

The server carries out causality (i.e., ordinal relation in this paper) control [10] over received MUs. The causality control is required to maintain the temporal order of manipulation events[†]. For the control, each haptic MU has a time limit which is equal to the generation time of the MU plus Δ milliseconds. If the MU is received by the server before the time limit, it is held in the buffer by the time limit, and it arranges them 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 N clients, the server would always use N 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 N MUs to be used for the calculation every millisecond.

To obtain the position, the server judges whether the PHANToM cursor touches each object. Then, the server calculates the force against the object by using a spring-damper model [9]. It multicasts the positional information as an MU to all the clients. The MU also includes the positional information of the cursors at all the clients.

When each client receives an MU, the client updates the positions of objects after carrying out media synchronization control and calculates the reaction force applied to the player by using GHOST (General Haptic Open Software Toolkit) [9]. The rendering rate of the haptic media is 1 kHz (i.e., the force is calculated at a rate of 1 kHz) at the client. Also, the redering rate of the virtual space is 30 Hz.

In order to adjust the output timing of haptic MUs among all the clients, we enhance the *virtual-time rendering* (*VTR*) algorithm [11] which employs the synchronization maestro scheme [6]. The scheme uses the synchronization maestro as shown in Fig. 1. The synchronization maestro gathers the information about the output timing of haptic MUs from each client, and it determines the reference output timing and multicasts the information about the reference output timing to all the clients. Each client gradually adjusts its output timing to the reference output timing[†††].

[†]How the causal relation is important depends on applications.

[††]This may lead to a disturbance of the causality. By exerting the Δ-causality control [10], under which the MU is discarded if it misses its time limit, we can preserve the causality. However, by experiment, we have confirmed that the Δ-causality control causes serious deterioration in the output quality of haptic media when the network load is heavy. This is because a number of MUs are discarded in this case.

[†††]For simplicity, we here assume that the timers of each client and the server are globally synchronized with each other. We can adjust the timers' values to each other by using the Network Time Protocol (NTP) or GPS.

Figure 1 also shows what kinds of functions the server and clients have. As shown in this figure, the media synchronization control is carried out at each client, and the server performs the causality control.

3. Group Synchronization Control Scheme

For group synchronization, as described earlier, we enhance the VTR algorithm which employs the synchronization maestro scheme [6]. The reason why we enhance the algorithm is that we make use of the servo loop of 1 kHz instead of the timer as described earlier. That is, the time is discrete in milliseconds in this paper. Note that an MU should be output every millisecond and the time resolution is 1 ms here. Therefore, we cannot exert the *shortening of output duration* or the *virtual-time contraction* [12], which is employed in the VTR algorithm; the shortening of output duration and the virtual-time contraction bring discarding MUs.

In order to explain the group synchronization control, let us focus on a client. We first define the *ideal target output time* [13] x_n of the n-th MU ($n = 1, 2, \cdots$) as the time at which the MU should be output in the case where there is no network delay jitter. 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* [10], which is defined as the time from the moment an MU is generated until the instant the MU should be output, and $\delta \leq \Delta_{\text{al}}$. We employ the *maximum allowable delay* Δ_{al} [13] in order to preserve the interactivity of haptic media.

We cannot always output each MU at its ideal target output time since there exists network delay jitter. Therefore, we next introduce the *target output time* [11] t_n of the n-th MU, which is calculated by adding some amount of time (called the *total slide time* [13]) to the ideal target output time.

Let us define the *slide time* [13] of the n-th MU, which is denoted by ΔS_n, as the difference between the *modified target output time* [11] 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* [10] in this paper, 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* [10] here. 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 this section.

In addition, the client notifies the synchronization maestro of the total slide time S_n whenever the target output time is modified [11] at the n-th MU (that is, in the case of the *virtual-time expansion* [12])($n \geq 2$).

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; in this paper, we handle Methods 1 and 2 to determine the reference value as described in Sect. 1. Then, the maestro multicasts the information about S to all the clients at regular intervals (every 5 seconds in our experiment in Sect. 4) [10].

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. The control in the case of $S = S_{n-1}$ is the same as that in [10], while we enhance the control in the other case for haptic media. The reason why we enhance the control is as follows. 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. This is because the output duration of each MU is 1 ms and we output only one MU every millisecond; therefore, the increase and decrease of the total slide time (i.e., the virtual-time expansion and contraction) bring pausing and skipping MUs, respectively [14].

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 d_n of the n-th MU as follows: $d_n = t_n$ (in this paper, since the server multicasts a single haptic media stream to each client, we do not need to perform inter-stream synchronization control. 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: $\Delta S_n = \min(r_1, T_n + \Delta_{\mathrm{al}} - t_n)$. 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 selected 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 [10]. 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. 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 $d_n = t_n^*$ [13]; 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 [10]; that is, the virtual-time contraction occurs. 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.

4. Experimental System

For simplicity, we set the number of players to two (i.e., $N = 2$) in the experiment. Each of the two players who have almost the same skill lifts and moves his/her object (a rigid cube) so that the object contains the target (a sphere) in a 3-D virtual space (height: 89.7 mm, width: 129.7 mm, depth: 89.7 mm) as shown in Fig. 2. Each side of the cube is a quarter of the virtual space's height, and the radius of the sphere is half of the cube's side. The mass of the object is 0.5 kg, and the acceleration of gravity is 2.0 m/s^2. The objects and target do not collide with each other, and the PHANToM cursors do not collide with 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 players compete on the number of eliminated targets with each other.

The server judges which object contains the target. If the distance between the center of the object and that of the target is less than 4 mm, the server judges that the object contains the target. Then, the server updates the position of the target and transmits the information about the position to the clients. Each client deletes the old target and displays a new target at the position.

4.1 System Configuration

As shown in Fig. 3, the experimental system consists of the server (CPU: Pentium4 processor at 2.26 GHz, OS: FreeBSD 4.7), clients 1 and 2 (CPU: Pentium4 processor

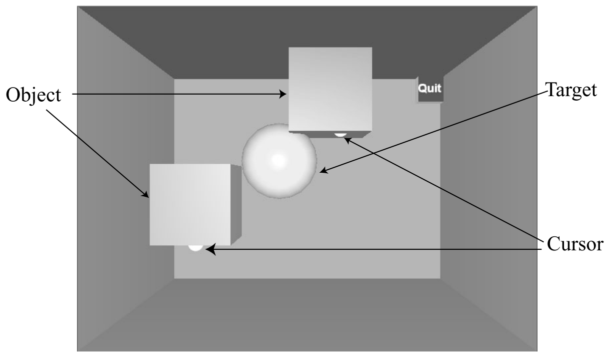


Fig. 2 A displayed image of the virtual space.

[†]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 [11].

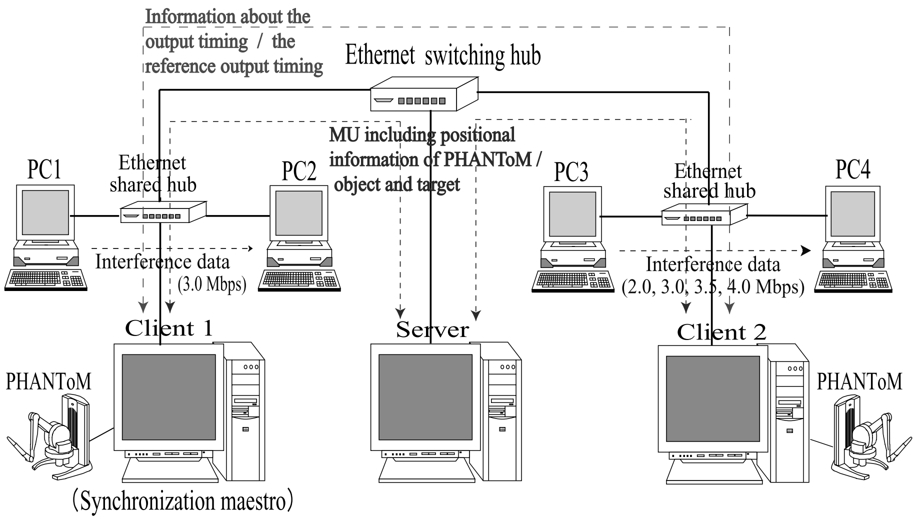

Information about the output timing / the reference output timing
Ethernet switching hub
MU including positional information of PHANToM / object and target
PC1
Ethernet shared hub
PC2
PC3
Ethernet shared hub
PC4
Interference data (3.0 Mbps)
Interference data (2.0, 3.0, 3.5, 4.0 Mbps)
Client 1
Server
Client 2
PHANToM
PHANToM
(Synchronization maestro)

Fig. 3 Configuration of the experimental system.

at 2.80 GHz, OS: WindowsXP), and four PCs (PC1 through PC4). They are connected to an Ethernet switching hub and two Ethernet shared hubs (10BASE-T). For simplicity, client 1 has a function of the synchronization maestro (for the influence of the position of the synchronization maestro, the reader is referred to [15]).

The size of an MU from the server to each client is 113 bytes, and that in the opposite direction is 33 bytes (since the former MU includes the positional information of the PHANToM cursors, the target, and the objects, the MU size is larger than the latter MU size). MUs and the information about the output timing/the reference output timing for the group synchronization control are transmitted by UDP. The size of the information from the maestro to each client is 3 bytes, and that from each client to the maestro is 5 bytes (the difference of 2 bytes corresponds to the number of bytes representing the client's ID).

In order to generate traffic flows of interference, PC1 (PC3) sends fixed-size data messages of 1472 bytes each to PC2 (PC4) at exponentially distributed intervals (see Fig. 3). For transmission of interference data, we also use the UDP protocol. The switching hub is employed so that the traffic flow of interference in one of the shared hubs does not affect the other. In the experiment, we set the *data load*, which is defined as the average number of interference data bits transmitted in a second, to 3.0 Mbps at PC1, and we select the data load at PC3 from among 2.0, 3.0, 3.5, and 4.0 Mbps.

This paper deals with *No group-sync* as well as Methods 1 and 2 . In No group-sync, each client exerts only the intra-stream synchronization control, which is also carried out in Methods 1 and 2. The server performs the same causality control in the three schemes.

In the experiment, we set the *maximum allowable delay* Δ_{al}, which is defined in the VTR algorithm [14], to 40 ms. Other parameter values were set to the same as those in [6]; for example, $\delta = 10$ ms, $T_{\mathrm{h2}} = 20$ ms, $r_1 = 5$ ms, $r_2 = r_3 = 3$ ms, $r_4 = r_5 = 1$ ms, $N_{\mathrm{c}} = N_{\mathrm{d}} = 1000$, and $N_{\mathrm{e}} = N_{\mathrm{f}} = 20$.

4.2 Performance Measures

As performance measures, we introduce the *elimination rate of the targets*, the *average number of eliminated targets*, and the *average total number of eliminated targets*. We also adopt the *mean square error of group synchronization* [6].

The elimination rate of the targets is defined as the ratio of the number of targets which are contained and eliminated by each client to the total number of appeared targets. The measure is related to the fairness. Since the two players have almost the same skill, the elimination rate of approximately 0.5 implies the fairness between the two players. The average number of eliminated targets is the mean number of targets which are contained and eliminated by each client at the client. The average total number of eliminated targets is the mean of the total number of targets which are eliminated by either of the two clients. The mean square error of group synchronization is the mean square of the difference between the output time of each MU at client 1 and that of the MU at client 2. As for group synchronization, how large time difference between the two clients is allowable is not clear; this is for further study [16].

The two players conducted the experiment 20 times at each of data loads considered here in each scheme (in total, 240 experimental runs). The measurement of the performance was carried out for 30 seconds from 5 seconds after the beginning of each experimental run[†]. We plot the averages of the obtained results in the figures to be shown in the next section. We also display the 95% confidence intervals.

5. Experimental Results

We show the elimination rate of the targets at client 1, the average numbers of eliminated targets at clients 1 and 2, and

[†]We lifted and moved the cube from the floor to the target within the 5 seconds.

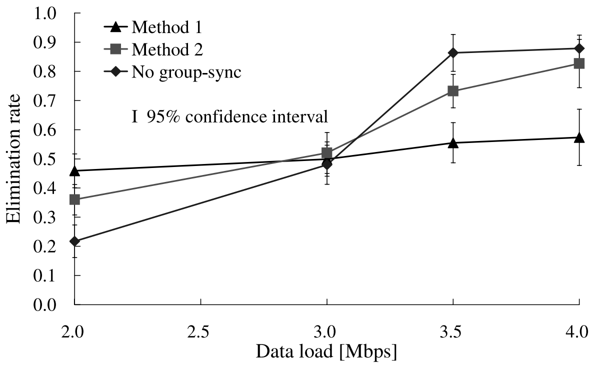


Fig. 4 Elimination rate of the targets at client 1 versus data load.

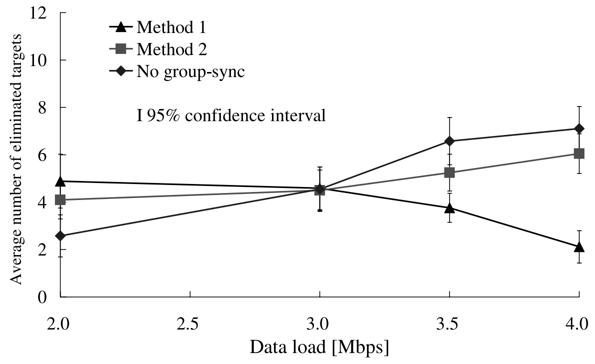


Fig. 5 Average number of eliminated targets at client 1 versus data load.

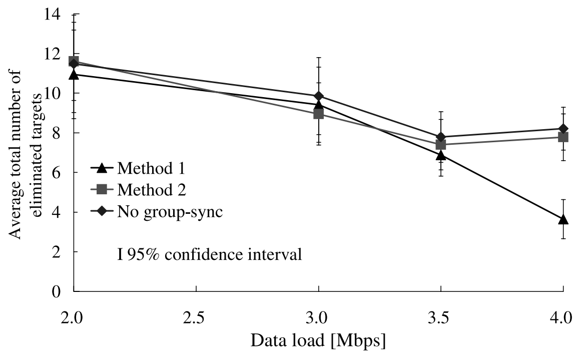


Fig. 6 Average total number of eliminated targets versus data load.

the mean square error of group synchronization as a function of the data load at PC3 in Figs. 4 through 7, respectively.

In Fig. 4, we see that the elimination rate of Method 1 is around 0.5 independently of the data load. Therefore, Method 1 is superior to the others in terms of the fairness. We also find in the figure that Method 2 is the second best, and No group-sync is the worst. Furthermore, when the data load is 3 Mbps, the elimination rates of the three schemes are almost the same. This is because the data load at PC1 is equal to that at PC3.

From Figs. 5 and 6, we confirm that in Method 1, the average number of eliminated targets at client 1 is around half of the average total number of eliminated targets in the whole range of the data load. This is the reason why Method 1 is the best in Fig. 4. In Fig. 6, we find that the average total number of eliminated targets of Method 1 decreases as the data load becomes heavier. Those of No group-sync and Method 2 do not decrease when the data load increases from 3.5 Mbps to 4.0 Mbps. This is because the numbers of eliminated targets of the two schemes at client 2 did not decrease largely in this area.

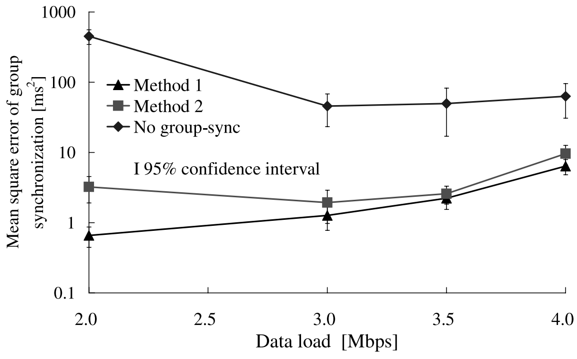


Fig. 7 Mean square error of group synchronization versus data load.

Figure 7 reveals that the mean square errors of group synchronization of Methods 1 and 2 are smaller than that of No group-sync in the whole range of the data load considered here. This is the effect of the group synchronization control. We also notice in the figure that Method 1 has slightly smaller mean square errors than Method 2.

From the above considerations, we can say that Method 1, which selects the later output timing as the reference output timing, is superior to the others in terms of the fairness[†].

6. Conclusions

This paper investigated the effects of group synchronization control for haptic media in a networked real-time game. We made a performance comparison among three schemes: Methods 1 and 2, and No group-sync. As a result, we saw that Method 1, which selects the later output timing as the reference output timing, is superior to the others in terms of the fairness.

As the next step of our research, we plan to handle the case of more than two players. We also need to carry out subjective assessment in order to clarify how large time difference among clients is allowable. Furthermore, we will study some networked games which require mutual interaction more strongly and other applications in which fairness is important. We will investigate the effects of the mutual interaction on group synchronization and fairness.

[†]We can use other output timings as the reference output timing. As an example, we used the mean of the output timing of the two clients in the experiment (we here call this method *Method 3*). As a result, we confirmed that the elimination rate of Method 3 is between those of Methods 1 and 2. Therefore, Method 3 is inferior to Method 1 and superior to Method 2 in terms of the fairness. We also need to deal with other output timings. This is worthy of further study.

Acknowledgment

The authors are grateful to Prof. Shuji Tasaka for his valuable discussions. This work was supported by the Grant-In-Aid for Scientific Research (C) of Japan Society for the Promotion of Science under Grant 16560331.

References

[1] M.A. Srinivasan and C. Basdogan, "Haptics in virtual environments: Taxonomy, research status, and challenges," Comput. Graph., vol.21, no.4, pp.1393–1404, April 1997.

[2] J.P. Wilson, R.J. Kline-Schoder, M.A. Kenton, and N. Hogan, "Algorithms for network-based force feedback," Proc. Fourth PHANToM Users Group Workshop, Nov. 1999.

[3] G. Blakowski and R. Steinmetz, "A media synchronization survey: Reference model, specification, and case studies," IEEE J. Sel. Areas Commun., vol.14, no.1, pp.5–35, Jan. 1996.

[4] Y. Ishibashi and S. Tasaka, "A group synchronization mechanism for live media in multicast communications," Conf. Rec. IEEE GLOBECOM'97, pp.746–752, Nov. 1997.

[5] Y. Ishibashi and S. Tasaka, "A distributed control scheme for group synchronization in multicast communications," Proc. ISCOM'99, pp.317–323, Nov. 1999.

[6] Y. Ishibashi, T. Hasegawa, and S. Tasaka, "Group synchronization control for haptic media in networked virtual environments," Proc. 12th IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (Haptics'04), pp.106–113, March 2004.

[7] J.K. Salisbury and M.A. Srinivasan, "Phantom-based haptic interaction with virtual objects," IEEE Comput. Graph. Appl., vol.17, no.5, pp.6–10, Sept./Oct. 1997.

[8] S. Matsumoto, I. Fukuda, H. Morino, K. Hikichi, K. Sezaki, and Y. Yasuda, "The influences of network issues on haptic collaboration in shared virtual environments," Proc. Fifth PHANToM Users Group Workshop, Oct. 2000.

[9] SensAble Technologies, "GHOST SDK programmer's guide," Version 3.0, 1999.

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

[11] Y. Ishibashi and S. Tasaka, "A synchronization mechanism for continuous media in multimedia communications," Proc. IEEE INFOCOM'95, pp.1010–1019, April 1995.

[12] Y. Ishibashi and S. Tasaka, "A comparative survey of synchronization algorithms for continuous media in network environments," Proc. IEEE LCN'00, pp.337–348, Nov. 2000.

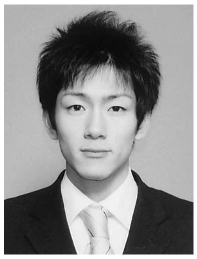
[13] S. Tasaka, T. Nunome, and Y. Ishibashi, "Live media synchronization quality of a retransmission-based error recovery scheme," Conf. Rec. IEEE ICC'00, pp.1535–1541, June 2000.

[14] Y. Ishibashi, S. Tasaka, and T. Hasegawa, "The virtual-time rendering algorithm for haptic media synchronization in networked virtual environments," Proc. 16th International Workshop on Communications Quality & Reliability (CQR'02), pp.213–217, May 2002.

[15] T. Nunome and S. Tasaka, "Inter-destination synchronization schemes for continuous media multicasting: An application-level QoS comparison in hierarchical networks," IEICE Trans. Commun., vol.E87-B, no.10, pp.3057–3067, Oct. 2004.

[16] H. Kaneoka and Y. Ishibashi, "Subjective assessment of fairness among players in networked game using haptic interface devices," IEICE Technical Report, CQ2005-30, July 2005.

Yutaka Ishibashi received the B.S., M.S., and Ph.D. degrees from Nagoya Institute of Technology, Nagoya, Japan, in 1981, 1983, and 1990, respectively. From 1983 to 1993, he was with NTT Laboratories. In 1993, as an Associate Professor, he joined Nagoya Institute of Technology, in which he is now a Professor in the Department of Computer Science and Engineering, Graduate School of Engineering. From June 2000 to March 2001, he was a Visiting Professor in the Department of Computer Science and Engineering at the University of South Florida. His research interests include networked multimedia applications, media synchronization algorithms, and QoS control. Dr. Ishibashi is a member of the IEEE, ACM, Information Processing Society of Japan, the Institute of Image Information and Television Engineers, and the Virtual Reality Society of Japan.



Hiroyuki Kaneoka received the B.S. degree from Nagoya Institute of Technology, Nagoya, Japan, in 2004. He is now a graduate student at the Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology. He is engaged in research on QoS control of haptic media in distributed virtual environments at Nagoya Institute of Technology.