A SCHEME FOR HAPTIC DATA TRANSMISSION UNDER VARIOUS NETWORK CONDITIONS

Yonghee You, Mee Young Sung, Kyungkoo Jun

Department of Computer Science & Engineering, University of Incheon
Department of Multimedia Systems Engineering Incheon 402-749, South Korea
{yhinfuture,mysung,kjun}@incheon.ac.kr

ABSTRACT

Haptic Collaboration Virtual Environment (HCVE) is an enhanced virtual reality space that supports sense of touch, which is called "haptic". In HCVE, remote users connected over networks are able to collaborate by sharing touching experiences in addition to well-established audio and visual interfaces. The success of HCVE largely depends on timely transmission of haptic data despite time-varying network conditions such as delay, loss, and jitter. However, the fact that the data generation frequency of haptic interface devices is extremely high, e.g. 1 KHz, makes the realization of successful HCVE more challenging. For seamless haptic data communication even under adverse network conditions, we propose a linear prediction algorithm and a buffering scheme. The prediction algorithm, which is basically extrapolation, is to mitigate the negative effects of network delay, loss and jitter, and the buffering scheme is to help synchronization of haptic interaction between remote users. We build an experimental test bed for the evaluation of our proposed schemes, and as the results of analyzing quantitative measurement results, conclude that those are effective in improving the quality of haptic experiences.

1. INTRODUCTION

"Computer Haptics" deals with the techniques for generating and displaying stimuli to the human user. Although there have been several recent studies focused on the development of multimodal virtual environments (VEs), less attention has been paid to human-human and human-machine interactions in shared virtual environments (SVEs) [1]. Haptic Collaborative Virtual Environment (HCVE) is a shared virtual reality space with haptic interface support. Users of HCVE collaborate each other over the network by using the haptic interfaces as well as audio and visual interfaces. With improved reality and immersion experience, HCVE is particularly proper for educational simulations [2].

Time-varying network conditions pose challenges to successful communication of haptic data [4]. Adverse network links sometimes cause irregular force-feedback which deteriorates the haptic experiences [3]. The transmission of the haptic data, which mainly consists of the position information of haptic device pointers, is basically similar to the multimedia streaming. However, it is much more demanding because the haptic rendering rate required for satisfactory haptic experience is quite higher than that of graphic rendering; 1 KHz for haptics, in contrast to 30 Hz for graphics. To meet such challenges, there have been various research efforts. For group synchronization control, Y. Ishibashi et al propose virtual time rendering algorithms [5]. Hikichi et al employ a queue monitoring algorithm [6] designed for efficient haptic collaboration. However, these approaches have limitations that they are not able to cope with delay, loss and jitter at the same time.

In this paper, we propose a scheme for the haptic data transmission to mitigate negative effects from delay, loss and jitter. The main idea is based on prediction and buffering. The prediction is to overcome the loss and the excessive delay, while the buffering is to keep moderate delay and jitter. We build an experimental HCVE test bed to evaluate the proposed schemes, and perform tests under different delay, loss and jitter conditions.

In section 2, our haptic collaboration application will be described. In section 3, the details of our proposed schemes are presented, the experiment results will be discussed in section 4, and we will conclude our work in the last section.

2. HAPTIC COLLABORATION APPLICATION

The development of our haptic collaboration application will be described in this section. The application enables human-to-human haptic interaction over the Internet. We developed a haptic collaborative system as shown in Figure 1. The objective is that users work together to move the cube to where the sphere is by 'haptically' touching the cube with their probes. The probes, one probe for one user, are drawn as small balls in the figure and represent the haptic interface pointers. Users are able to lift, push, or rotate the cube if they manipulate the probes cooperatively: for example, to lift up the cube, at least two users should position their probes underneath the cube and move them upward without losing balance.

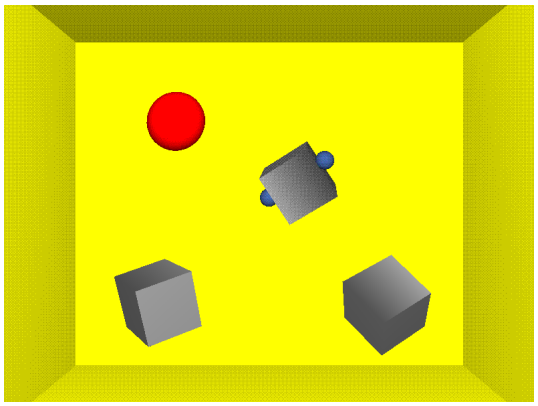

Figure 1. Snapshot of the haptic collaboration application

The aforementioned haptic application is developed with several software libraries. QUANTA networking library is used to implement the haptic data transmission over UDP. For haptic rendering, we use OpenHaptics toolkit [7] which provides both Haptic Library API(HLAPI) and Haptic Device API (HDAPI). HLAPI is for rendering haptically static objects such as the room in Figure 2 and able to generate high quality force-feedback based on OpenGL frame-buffer. HDAPI is for rendering dynamic objects such as the cube. Open Dynamics Engine (ODE) is also used to enhance the movement animation of the cube.

The haptic collaboration is based on the client and server architecture as shown in Figure 2: clients send their own haptic data to a server, which in turn performs calculation necessary for the haptic rendering. The haptic data of clients, mainly haptic pointer positions, are obtained from haptic interfaces attached to client machines. With the received haptic position data, the server detects possible contacts between the client haptic pointer and the cube, and applies the spring-damper model to obtain positions and rotation angles of the cube.

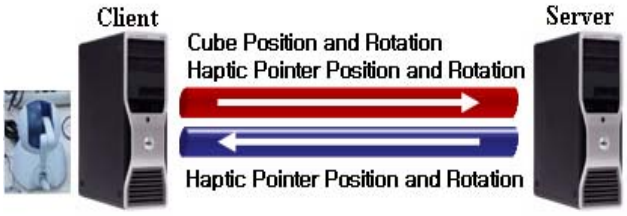

Figure 2. Client/Server Architecture

3. HAPTIC DATA TRANSMISSION SCHEME

In this section, we propose a scheme for the haptic data transmission. Considering that the haptic data is susceptible to the network conditions [8], the basic idea of our scheme is the well-balanced combination of dead-reckoning, buffering and synchronization as shown in Figure 3.

Before discussing our scheme in detail, we first introduce some of supporting modules to compose our scheme. Calculation Module (CM) of the server is responsible for the collision-detection and physics calculation, Network Modules (NMs) of the client and the server are for communicating the haptic data in UDP, Rendering Module (RM) of the client is for graphic rendering, and Haptic Input Module (HIM) of the client is for collecting data from the haptic interfaces.

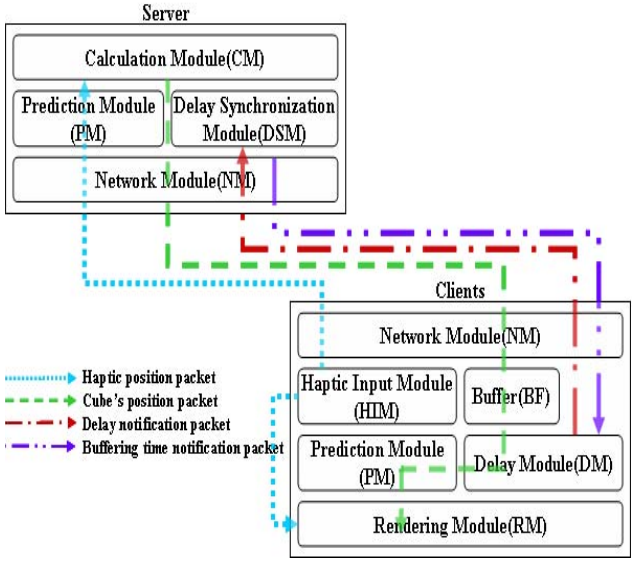

Figure 3. Scheme for Haptic Data Transmission

The most important parts of our proposed scheme are Prediction Module, Delay Synchronization Module, Buffer, and Delay Module. We discuss each of the modules in detail in the following subsections.

Prediction Module (PM) is to compensate for packet loss and jitter. Since clients and a server transmit packets every 1 *ms*, PMs on both the clients and the server check whether packets arrive every 1 *ms*. If packet loss is detected, PM applies a linear algorithm to predict next positions of the haptic pointers and the cube. In the case of jitter, out-of-order packets are detected by packet sequence numbers and discarded. PM compensates for the discarded packets in the similar way to the lost packet compensation.

The position prediction is executed as follows:

$$x_n = x_{n-1} + v \tag{1}$$

$$v = \frac{\sum_{n=1}^{k} (x_{n-k} - x_{n-(k+1)})}{k-l} \tag{2}$$

where x_n is the predicted position while x_{n-1} corresponds to the previous position, v is the velocity of the cube or haptic pointer's position. The velocity v is computed by averaging the previously received positions' velocities. k is

the number of previously received positions involved in the computation. *l* indicates the number of lost positions among the position included in the computation of *v*. However, *l* is ignored in the normal version, and only calculated in the extended version.

Delay disturbs the haptic collaboration. One example of the negative effects of delay is that, in the case of the application of Section 2, all clients see the same cube but all in different positions.

In order to cope with this spatial de-synchronization problem, we propose Delay Synchronization Module (DSM) at the server side and Delay Module (DM) at the client side. The whole idea of DSM and DM for synchronization is that clients should put off their haptic rendering until all of them become ready to start the rendering simultaneously. For this, DSM decides *buffering times* for each client, i.e. how much time each connected client should wait before its rendering. Clients are required to report their measured delay to the server every 5 seconds and DSM calculates the buffering times for each client as follows.

$$M = MAX(D_1, D_2, D_3, ..., D_n) \quad (3)$$

$$R_n = M - D_n \quad (4)$$

where D_i is the delay reported from client *i*, *M* is the maximum of *Di*, R_i is the buffering time for client *i*. For example, if *M* is 100 *ms* and D_1 is 10 *ms*, the buffering time for client 1 becomes 90 *ms*.

During the buffering time, clients store the received haptic packet data into their Buffer (BF). Since every packet has its sequence number, the packets in BF can re-ordered in sequence, thus eliminating jitter effect. When starting the haptic rendering with the data in BF, DM uses the following equations to calculate the haptic positions.

$$P_n = C - R_n (R > 0) \quad (5)$$

$$CP_n = BF[I(P_n)] \quad (6)$$

where *C* denotes the current time in *ms*, R_n denotes the buffering time, P_n is the difference between the current time and the buffering time, $I(P_n)$ is the index of the buffer, and CP_n is the position data that will be rendered.

4. EXPERIMENTS

We perform experiments to evaluate the efficiency of our proposed scheme. In this section, we describe a test bed, assessment methods and discuss results.

The test bed is composed of three parts; a server, clients, and a network emulator, NistNet, which is able to emulate a wide variety of network conditions [9]. The haptic device that we use for experiments is PHANToM Omni [7]. We set its rendering rate to 1 KHz. Other details are shown in Table 1.

Table 1 Test bed Configuration

Server Computer	Client Computer
- AMD AthlonTM 64 Processor 3500+ 2.21GHz 1.00 GB RAM - OS : Microsoft Windows XP Professional Version 2002 Service Pack2	- Sensable PHANToM Haptic device - Dell Precision PWS380 Intel® Pentium 4 CPU 3.20GHz, 1.00GB RAM - NVIDIA Quadro FX 1400 Graphic Card - OS : Microsoft Windows XP Professional Version 2002 Service Pack2

To evaluate the performance of PM, we compare the cube positions obtained from the clients with the predicted positions at the server under various network conditions of delay, loss, and jitter.

For DSM and DM, we record the cube positions at each client and compare them each other to see the differences. The expression used for evaluation is as follows:

$$e_i = \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2} \quad (7)$$

where x_i, y_i, z_i are the current position of the cube at client *i* , while x_i', y_i', z_i' are the latest cube position calculated at the server. The smaller e_i is, the more accurate our scheme is.

In the first experiment, we evaluate PM under 15% packet loss condition. We set *k* in PM as 1, 3, 5 in the normal version and 4 in the extended version in order to find out the most appropriate equation for PM and also perform the experiment without the scheme. As shown in Figure 4 where *x* axis indicates the elapsed time in millisecond and the *y* axis represents errors. Our proposed scheme shows outperforms the rest when *k* is 5 in extended version. The average errors on conditions *k*=1, 3, 5 in normal version and 5 in extended version are 0.109, 0.093, 0.060 and 0.054 respectively while 0.989 in the experiment without PM. When *k*=1, the average error is higher than the experiment without PM, because PM is not able to deal with consecutive loss.

In the second experiment, we evaluate PM under 50 *ms* jitter. As shown in Figure 5, PM with *k*=1 in extended version is still the most accurate among others. The average errors on conditions *k*=1, 3, 5 in normal version and in extended version are 2.745, 0.434, 0.446 and 0.401 while 0.542 without PM. Note that the average error in the case of *k*=1 is much higher than in the case of others. This is also because PM cannot handle more than *k* consecutive lost packets.

In the third experiment, we evaluate DSM and DM. We install one server and two clients, and apply 100 *ms* delay to the link between the server and one of the clients so that two clients are in asynchronous state. We calculate the differences between the positions of two clients using our proposed scheme with *k*=1, 3, 5 and without the scheme respectively. In Figure 6, the circle plotted line represents the differences between the positions of two clients under 100 *ms* delay. The square plotted line represents that of two

clients without delay. The *y* axis represents the differences between the cube's positions of the server and of the client. As the results, we observe that our scheme is still able to achieve better performance than the case without the scheme. As shown in Figure 6, without our scheme, the value of the circle plotted line is quite high and the haptic collaboration is hardly achievable. However, when our scheme is activated, the differences are negligible.

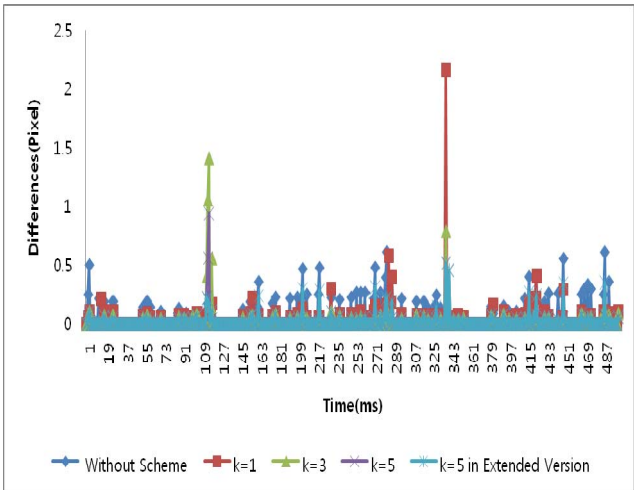

Figure 4. Differences under average 15% packet losses

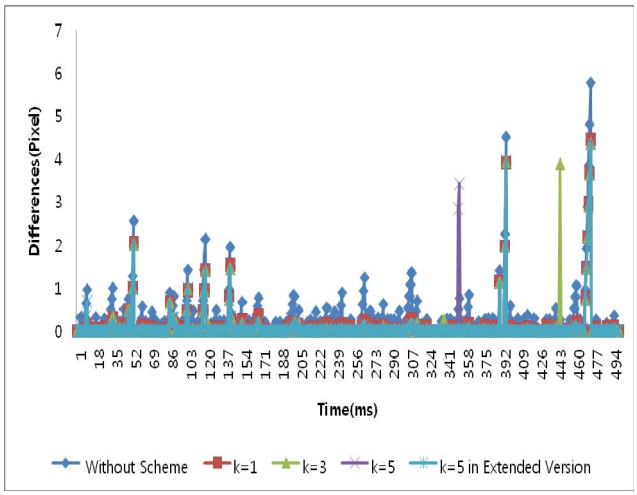

Figure 5. Differences under average 50 *ms* jitters

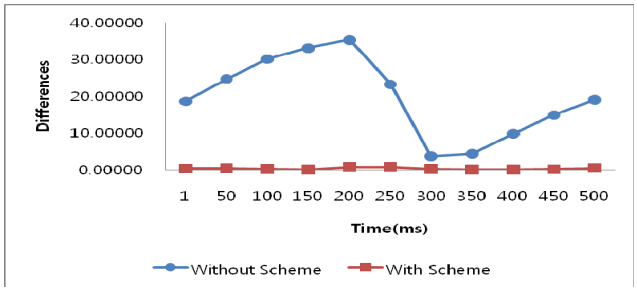

Figure 6. Differences under 100 *ms* delay

5. CONCLUSION

The objective of our study is to develop a haptic data communication scheme that contributes to the successful implementation of HCVE. In order to mitigate the negative effects of delay, loss, and jitter conditions even under unstable real network conditions, we propose the linear prediction and buffering scheme. The prediction algorithm is basically extrapolation and it mitigates the negative effects of network delay, loss and jitter. The buffering scheme helps the synchronization of haptic interaction between remote users. In the experiments, we evaluate the performance of our proposed schemes and find that the results are promising.

6. ACKNOWLEDGEMENT

This work was supported by the Brain Korea 21 Project in 2007, by grant No. RTI05-03-01 from the Regional Technology Innovation Program of the Ministry of commerce, Industry and Energy (MOCIE).

7. REFERENCES

[1] M. A. Srinivasan, and C. Basdogan, "Haptics in virtual environments: Taxonomy, Research status, and Challenges," Computers and Graphics, 1997.

[2] Hosseini, Mojtabe et al. "A Haptic Virtual Environment for Industrial Training". In Proceedings of HAVE 2002 IEEE International Workshop on Haptic Audio Visual Environments and their Applications, IEEE, Ontario, Canada, November 2002

[3] M. O. Alhalabi, S. Horiguchi, and S. Kunifuji, "An experimental study on the effects of network delay in cooperative shared haptic virtual environment," Computer and Graphics, vol. 27, pp. 205-213, 2003

[4] K. Hikichi et al. "Architecture of Haptics Communication System for Adaption to Network Environments", IEEE International Conference on Multimedia and Expo Proceedings, pp. 744-747, 2001

[5] Y. Ishibashi, T. Hasegawa, and S. Tasaka, "Group Synchronization Control for Haptic Media in Networked Virtual Environments," Proceedings of the 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004

[6] D. L. Stone and K. Jeffay, "An empirical study of delay jitter management policies," ACM Multimedia Systems, Vol. 2, pp. 267-279, Jan. 1995

[7] PHANTOM-Omni http://www.sensable.com/

[8] A. Boukerche, S. Shirmohammadi and A. Hossain "A Prediction Algorithm for Haptic Collaboration" Proceedings of HAVE 2005, pp. 154-158. October 2005

[9] Nist-Net : http://www-x.antd.nist.gov/nistnet/