HoIP: Haptics over Internet Protocol

Vineet Gokhale
Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai
vineet@ee.iitb.ac.in

Onkar Dabeer
School of Technology and Computer Science
Tata Institute of Fundamental Research
Mumbai
onkar@tcs.tifr.res.in

Subhasis Chaudhuri
Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai
sc@ee.iitb.ac.in

***Abstract*—Applications such as telesurgery need low latency communication of haptic data between remote nodes. For the stability of the control loop, a typical round trip delay target is 5 ms or lower. In this paper, we specify an application layer protocol - Haptics over Internet Protocol (HoIP) - which is designed to allow such low latency haptic data transmission without sacrificing on the quality of haptic perception. The key ingredients of HoIP include adaptive sampling strategies for the digitization of the haptic signal, use of a multithreaded architecture at the transmitter and receiver to minimize processing delay, and use of an existing UDP implementation. In this paper, we describe in detail the frame structure used in HoIP and the rationale behind our design choices, the implementation details of HoIP at the transmitter and receiver, and report processing delay measurements done using real haptic devices and HAPI (an open source software used to interface with the haptic device). The HoIP software is entirely written in C++ and our results show that in the worst case transmitter and receiver side processing delays are less than 0.6 ms.**

Keywords- *Haptics, Weber's law, level crossings, extrapolation, application layer, latency*

I. Introduction

A wide variety of applications exploit tactile haptic feedback. These include teleoperation [1], telementoring and collaborative surgical simulations [2], medical diagnostics [3], cultural heritage museum [4], entertainment and gaming [5], etc. Such applications often involve multiple users interacting in a networked haptic virtual environment. This is referred to as a collaborative haptic audio visual environment (C-HAVE) [6], or telehaptics [7]. In interactive haptic applications, such as teleoperation, the quality of perception is determined by the closed loop delay. If the closed loop delay is larger than the coherence width of the haptic signal (which is the inverse of the haptic signal bandwidth), then actions applied by the teleoperator take effect only when the haptic signal has changed substantially. This leads to performance degradation and even instability; see for example [1]. Hence, we require very low latency, and prior works such as [8], suggest a target of 5 ms or lower. The round trip latency consists of the haptic force capture and packetization delay at the transmitter, the receiver packet processing and rendering delay, and the network delay. In this paper, we describe a software stack designed to minimize the sender and receiver processing delays. In the remainder of this paper, we describe prior work and summarize our main contributions.

In traditional voice over IP (VoIP) systems, it is common to use 20-30 ms voice frames and use signal compression techniques to form transmission packets at 30-50 packets/s [9]. The delay of considering signal blocks is considerable for haptic signals and this approach cannot be used. An alternative is to send samples as they are without any block processing. The input and output refresh rate of haptic devices is usually 1 kHz. Thus transmission of haptic signals by using one packet per sample in the naive case leads to 1000 packets/s. In order to reduce the packet rate, many researchers have proposed the use of adaptive sampling based on perceptually significant characteristics of the haptic signal such as Weber's law and level crossings; see for example [10], [11], [12], [13]. We use adaptive sampling along with UDP in our software stack.

A number of generic networking protocols are available for virtual environments and transmission of multimedia. Some examples are the Synchronous Collaboration Transport Protocol (SCTP) [14], the Reliable Multicast Transport Protocol (RMTP) [15], and the Real Time Protocol for Interactive Applications (RTP/I) [16]. Since these are not designed specifically for haptic applications such as teleoperation, they have some limitations - the focus on reliability in SCTP and RMTP comes at the cost of increased delay, while the overheads in RTP/I seem rather high. Consequently, some researchers have attempted to create protocols specially suited to haptic signals. The Application Layer Protocol for Haptic Networking (ALPHAN), proposed in [17], employs a separate buffer for the haptic signal and video. The updates are categorized as I, P and B. I packets are sent reliably, whereas P and B packets are encoded differentially based on previous and future frames. The network parameters are exchanged between systems using an XML parser that contains tags with the parameter values specified. The possibility of adaptively compressing the haptic data was totally unexplored in this protocol. Hence a huge traffic, at the typical rate of 1000 packets/s, is generated. The Perception based Adaptive Haptic Communication Protocol (PAHCP), proposed in [18], exploits Weber's law of perception to reduce the number of packets transmitted over the network. Due to the assumption of a fixed Weber threshold at both ends, adaptive sampling is not exploited to the best possible extent. In reality, the Weber threshold is a variable dependent on the individual as well as other factors such as the mood of the person. In general, this protocol seems difficult to tune to specific user cases. We are also not aware of any transmitter and receiver processing delay analysis for this work.

Funding supports from Indian Digital Heritage project, Bharti Centre for Communication and NPPE programs are gratefully acknowledged.

978-1-4799-0849-3/13/$31.00 ©2013 IEEE

In HoIP, the option of transmitting the data at the haptic loop rate or reduced rate based on adaptive sampling is given to the system administrator. Adaptive sampling is based on Weber's law and level crossings. There is also room to add other sampling schemes in the future. The sampling parameters are user specific since perception varies across users. Hence the adaptive sampling parameters are communicated from one end to the other periodically. This allows different human operators to use the same C-HAVE system and the packet transmission rate can be adapted to their perception abilities. HoIP leverages existing UDP and IP implementations and is written in C++. It is multithreaded to minimize transmitter and receiver side processing delays. We have used HoIP for networking haptic devices over a LAN and we report delay measurements. We find that the transmitter and receiver side delays are below 0.6 ms. Thus HoIP and our implementation has very low delay and can be used in networks with round trip delays less than about 4 ms to yield an overall round trip delay of 5 ms or less.

The paper is organized as follows. In Section II we describe the rationale behind various design choices made in HoIP. In Section III, we describe the building blocks and implementation details of the protocol. In Section IV-A, we describe the experimental setup, in Section IV-B we present the delay measurements, in Section IV-C we give examples of signals, their samples and their remote reconstruction, and in Section IV-D we study the average packet rates as a function of the sampling parameters. In Section V, we summarize our conclusions and list open directions.

II. HoIP Protocol

This section is organized as follows. Section II-A specifies the location of HoIP in the networking stack. In Section II-B, we discuss the adaptive samplers implemented as a part of HoIP. In Section II-C, we describe the need for a minimum transmission rate. In Section II-D, we specify the extrapolators used at the receiver. In Section II-E, the various fields of HoIP frame are described.

A. Application Layer Protocol

The HoIP protocol operates in the application layer. Since we are interested in real time applications, HoIP uses UDP as the transport layer protocol. Unlike TCP, which guarantees that a packet is sent successfully to the receiver, in UDP there is no assurance of successful packet transmission. However, TCP adds substantially to the network delay and for real time applications a delayed packet is often a useless packet. Hence UDP is preferred. In addition to packet losses, UDP also does not guarantee reception of packets in order. Hence packets need to carry time stamps or sequence numbers. We use the latter approach. Figure 1 shows the different protocols in the TCP/IP model and the position of HoIP.

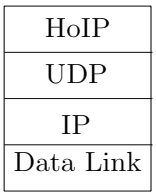

Figure 1 : HoIP in the protocol stack

B. Adaptive sampling

A first simple solution to digitizing haptic signals is to sample them at 1 kHz (the typical refresh rate of haptic devices) and send a packet for each sample. While in areas served with high speed internet connections the resultant packet rate of 1000 packets/s is manageable, many haptic applications are expected to operate in networks with limited bandwidth. Telesurgery in a disaster zone using quickly deployed networks is an example. Also, it is a good engineering practice to only generate as many data packets as needed and no more. Hence there is a need to reduce the packet rate compared to the naive solution. If we look at a typical VoIP system, then the packet rates are in the range of 30-50 packets/s. This is achieved by processing voice frames of size 20-30 ms at the transmitter. For haptics, given the round trip delay constraint of few ms, such frame based processing is not effective. Hence we resort to adaptive sampling - the haptic signal is sampled based on causal, perceptually significant features and a packet is transmitted for each perceivable sample. We refer to [13], [19] for a study of adaptive sampling of haptic signals. Based on these works, in HoIP we implement the following two adaptive samplers and we also leave room for implementation of other samplers in the future.

1) The Weber sampler, which uses the previous sample level X_{n-1} as a reference level and takes the next sample, X_n, at the first time t such that

$$\left|\frac{X(t) - X_{n-1}}{X_{n-1}}\right| \geq \delta. \quad (1)$$

2) The level crossings sampler, which uses the previous sample level X_{n-1} as a reference level and takes the next sample, X_n, at the first time t such that

$$|X(t) - X_{n-1}| \geq c. \quad (2)$$

Here δ is the Weber constant and c is the level crossing threshold. These constants can vary significantly depending upon the user and the haptic device [19]. Hence in HoIP we allow room for sending the adaptive sampler parameter value in the packet (see field *Threshold* in Table 1/Figure 2). The choice of the adaptive sampler is made at the start of a session and remains fixed during the session.

C. Minimum packet transmission rate

If the haptic signal changes only very slowly, then the reference of the previous sample used by the adaptive sampler is unreliable. Hence it is important to ensure that a minimum packet transmission rate is maintained. We ensure that the gap between successive packets is never more than 100 ms by periodically transmitting a packet with the current sample once in every 100ms, apart from transmitting the packets due to adaptive sampling, thus ensuring a minimum packet rate of 10 packets/s (or a minimum sampling rate of 10 Hz). This has an added advantage that if a packet is dropped (which is possible with UDP), then the signal drift at the receiver remains in control.

HoIP Version 1.0 is open source code. It can be downloaded from http://tinyurl.com/qdy5mod

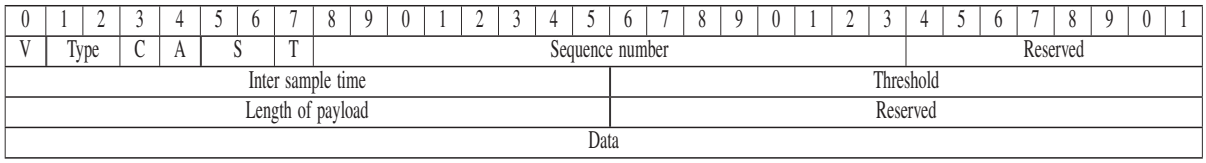

Figure 2: HoIP frame structure

D. Extrapolation at Receiver

The receiver needs to reconstruct a continuous time haptic signal for rendering based on the discrete packets received. To reduce delay, it uses previous information and extrapolates data till a new data packet is received. The HoIP protocol implements two extrapolators at the receiver. The user at the receiver has to choose one of the available extrapolators before reception could begin. The opening command prompt interface provides instructions regarding the procedure for choosing the extrapolator. At time t let the most recent samples received be X_{n-1}, X_{n-2} with corresponding received time stamps t_{n-1}, t_{n-2}. We use the following extrapolators in HoIP.

Sample-and-hold extrapolation: The signal rendered at time t is X_{n-1} for $t_{n-1} \leq t < t_n$ where t_n is the arrival time of the next packet.

Linear extrapolation: For level crossings, define

$$\underline{X}_{n-1} = X_{n-1} - c, \quad \bar{X}_{n-1} = X_{n-1} + c,$$

and for Weber's sampler, define

$$\underline{X}_{n-1} = (1-\delta)X_{n-1}, \quad \bar{X}_{n-1} = (1+\delta)X_{n-1}.$$

Let $L(a, b; x)$ be the soft limiter function that equals x if $a \leq x \leq b$, equals b for $x > b$ and equals a for $x < a$. Then the signal rendered at time t is

$$L\left(\underline{X}_{n-1}, \bar{X}_{n-1}; X_{n-1} + \frac{(t - t_{n-1})}{(t_{n-1} - t_{n-2})}(X_{n-1} - X_{n-2})\right)$$

for $t_{n-1} \leq t < t_n$ where t_n is the arrival time of the next packet. The role of the limiter is to ensure that we use linear interpolation only to the extent that it does not change perception as per the rule used for adaptive sampling. This ensures that if the reconstructed signal is passed through the same adaptive sampler, then we get the same samples as on the true signal.

While we implemented the above two extrapolations, HoIP has enough flexibility to support the implementation of additional extrapolators, if needed.

E. Frame Structure

The frame structure of HoIP is as shown in Figure 2. Each row in the frame is composed of 32 bits. The top row indicates bit positions of the fields. The bits are numbered from 0 to 9 for better readability. The HoIP frame starts from the second row. The frame consists of 12 bytes of header, out of which 3 bytes are reserved to accommodate future enhancements to the protocol like inclusion of video, audio data, etc. The purpose of the other 9 bytes are described in Table 1.

The fields *S, T, Inter sample time* and *Threshold* are read only if *A* is set to 1. These fields hold valuable information only if adaptive sampling is employed. The *Inter sample time* is 1 ms for typical haptic loop rate of 1kHz. The *Length of payload* depends on *Type* and *C*.

Field	Bits	Description
V	1	Protocol version set to 0 for haptic point to point communication (Current implementation)
Type	2	Indicates type of the data. Set to 0 for haptics, 1 for haptics-audio, 2 for haptics-video, 3 for haptics-audio-video
C	1	Indicates content type of the payload. Set to 0 if only force co-ordinates are transmitted, 1 for position and velocity co-ordinates
A	1	Indicates if adaptive sampling is applied or not. Set to 1 for adaptive sampling, 0 for haptic loop rate transmission
S	2	Indicates the type of adaptive sampling employed. Set to 0 for Weber, 1 for Level crossings. 1-bit is reserved for other samplers to evolve
T	1	Indicates transmission of threshold in the packet. Set to 1 if threshold is sent, else set to 0
Sequence number	16	Identifier of each packet to track lost or out of order packets
Inter sample time	16	Indicates the inter sample time gap. Along with the received times, this can be used to calculate packet delays
Threshold	16	Value of the threshold in percentage. If set to 0, data is transmitted at the haptic loop rate
Length of payload	16	Indicates length of the payload

Table 1: HoIP frame description

III. Implementation Details

The features described in the previous sections are implemented in C++. The C++ language is chosen over other programming languages for our real time application because of its faster execution in case of multithreaded programs. The different functionalities of HoIP are handled by multiple threads. The individual components employed in the protocol to implement the features of the protocol are described in this section. Figure 3 shows the high level architecture of HoIP and its different components are described next.

A. Named pipe

A named pipe, described in [20], is characterized by First-in-First-out (FIFO) data flow and is well suited for an application which requires data exchange between two C++ executables in the same system. For simplicity, we used half-duplex named pipes for sharing data between the haptic interface (in our case HAPI, discussed in [21], [22]) and HoIP. Each of the end systems have two pipes, one associated with sending the samples and the other with receiving the

samples. The usage of named pipes increases the ease with which HoIP can be employed for haptic communication, since a minimal modification in the haptic interface code will get the application working. This feature makes HoIP more plug-and-play when it comes to integrating HoIP and any haptic interface.

B. Sender module

The *Sender module* of HoIP reads samples from the named pipe, adaptively samples the data, packetizes the perceivable samples and finally sends the packets to the lower layer. These operations are managed by multiple threads. The *Pipe Reader Thread* keeps track of *Sender Pipe*, and the moment the pipe is loaded, this thread reads the sample and loads it in to the *Temporary Queue*. The transfer of a sample from the *Sender Pipe* to the *Temporary Queue* is done to make sure that the only bridge between the haptic interface and HoIP is not overloaded, which could otherwise result in loss of data. The *Adaptive Sampler Thread* keeps monitoring the *Temporary Queue* and subjects the samples to the thresholding test according to the appropriate equation (1) or (2). The samples, which are predicted to be perceivable, are loaded in to the *Sender Queue*, which is monitored by the *Packet Sender Thread* that packetizes the sample and sends it to the lower layer, which is UDP.

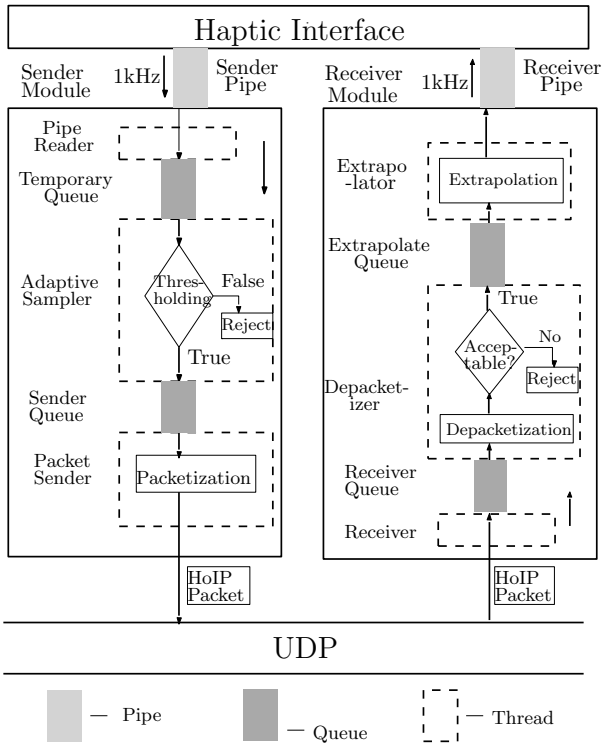

Figure 3 : Architecture of HoIP

C. Receiver module

The *Receiver module* of HoIP holds the responsibility of receiving packets from UDP, depacketizing it and loading the pipe for the *Haptic Interface*, which is used for rendering. The *Receiver Thread* monitors a dedicated socket for packets intended for the machine in which it is running and loads the packets in to *Receiver Queue*. The *Depacketizer Thread* extracts the sample by depacketizing the *Receiver Queue* contents. To ensue in-order delivery of packets, it also checks the sequence number to determine if a received signal sample is to be accepted or rejected. If the sample is acceptable, then it is loaded in to the *Extrapolate Queue*, which is monitored by the *Extrapolator Thread*. This thread implements the extrapolation based on the queue data and loads the *Receiver Pipe* with the predicted values at the haptic sampling rate which is usually 1kHz. The *Receiver Pipe* is constantly monitored by the *Haptic Interface*, which renders the haptic force.

Note: If the data transmission at haptic loop rate is chosen, then in the *Sender module* the *Adaptive Sampler Thread* is bypassed and the data from the *Temporary Queue* is directly packetized. In the receiver module, after the acceptability check, the data is transferred from the *Extrapolate Queue* to the *Receiver Pipe*.

IV. Measurements

A. Experimental Setup

In this section, we describe the experimental set up. The haptic devices used were the Phantom Omni ([23], [24]) with six degrees of freedom output capability, at both the ends. HAPI, which is an open source, cross platform haptic rendering software written in C++ was used. Both end systems, running on Windows, were connected on a 100Mbps LAN. A graphically rendered object was displayed at the operating end system. In order to get them acquainted with the system, the users were initially asked to explore the virtual environment for some time. The user holds the stylus of the haptic device for exploration and the remote collaborator was asked to feel the force. The thresholds are initially learned through a controlled experiment as discussed in [13]. Once the user is comfortable with the system, we begin our experiment of manipulating the virtual environment. This experiment was conducted 10 times with different users and the results reported are as shown in the next section.

B. Delay

The round trip delay of the signal is the sum of all processing delays and the network delay. Let RTT be the round trip time, T_{HS} be the HoIP sender delay, T_{HR} be the HoIP receiver delay, T be the delay due to lower layers (namely UDP packetization, IP packetization and data link layer delay at both sender and receiver), and let D be the overall network delay. Then the round trip delay is

$$RTT = 2(T_{HS} + T_{HR}) + T + D \tag{3}$$

Figure 4 shows the histogram of the HoIP sender delay. This includes reading data from the sender pipe, adaptively sampling and packetizing the data, and sending the HoIP packet, without the network delay. Figure 5 shows the histogram of the HoIP receiver delay. This includes reception of the packet, depacketization, checking for the acceptability of the packet, extrapolating the data and loading the receiver pipe. The above results show that the maximum HoIP sender delay is 0.45 ms, and maximum HoIP receiver delay is 0.13 ms. This makes the maximum round trip time using the Equation (3) to be $1.16 + T + D$ ms. Hence if $T + D$ is below 3.84 ms, then the target of bringing the overall delay below 5 ms is achieved.

It may also be noted that from Figures 4 and 5 that in both cases, the delay distributions do not show any sign of a long tail suggesting that the proposed protocol would rarely deviate from a predefined QoS if the network delay conditions are met. Also, the HoIP sender delay is independent of the time required in learning the sampling parameter of the operator.

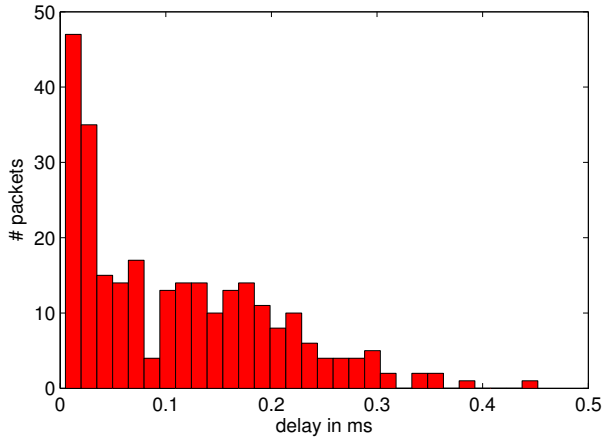

Figure 4 : Histogram of sender processing delay

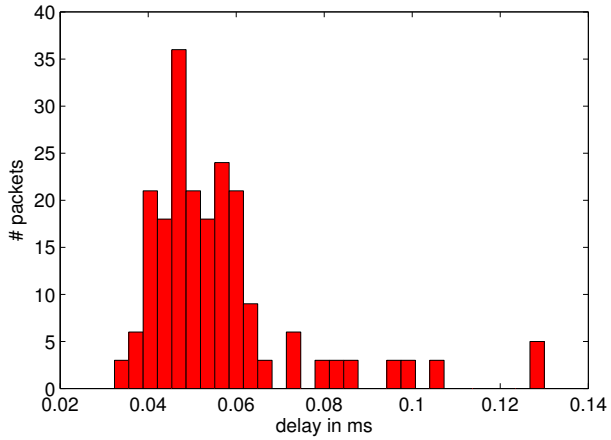

Figure 5 : Histogram of receiver processing delay

C. Signal Graphs

Figure 6 shows a comparison of the haptic signal sent over the network and the reconstructed signal at the operator, using a level crossings based sampler with a typical level crossings threshold of 0.22, as discussed in [13], [19], and sample-hold extrapolator at both ends. Figure 7 shows a comparison of the haptic signal sent and the reconstructed signal at the operator, using a Weber sampler with a typical Weber threshold of 0.12, as discussed in [13], [19], and linear extrapolator at both ends. For simplicity and clear analysis, one-dimensional signal is shown. The signals are plotted without adjusting for delay so that the overall delay is visible. The same haptic rendering program was run at both ends to test the working of different components of the protocol.

D. Average packet rate measurements

For measuring the packet rate of HoIP, we chose 5 values of thresholds, in the typical range, for each of the two samplers

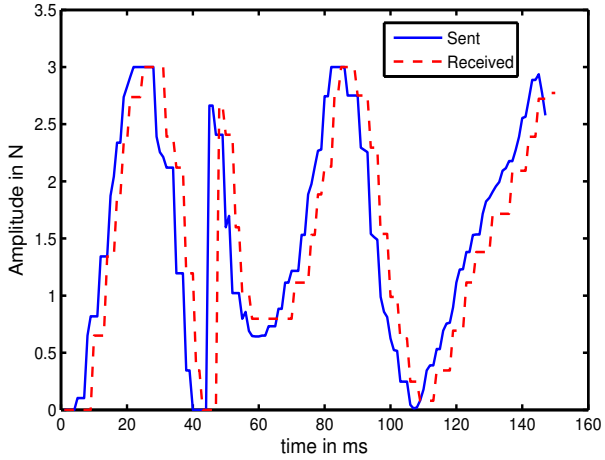

Figure 6 : Example of a typical sent haptic signal and the corresponding adaptively thresholded and extrapolated receiver output for a level crossings threshold of 0.22 and a sample-hold extrapolator

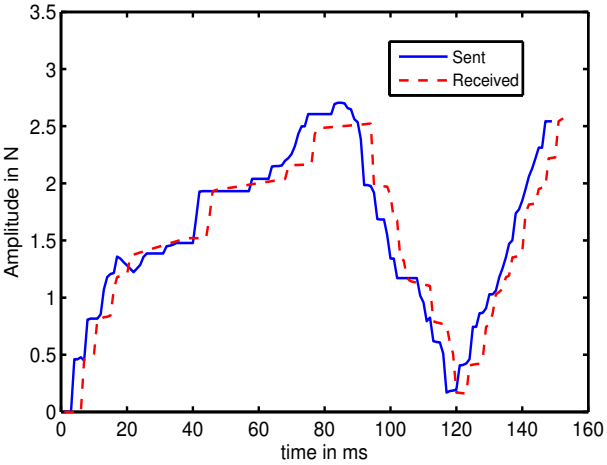

Figure 7 : Example of a typical sent haptic signal and the corresponding adaptively thresholded and extrapolated receiver output for a Weber threshold of 0.12 and a linear extrapolator

and the experiment, described in IV-A, was conducted 5 times, each lasting for 20 seconds. Finally the results for each threshold were averaged to obtain the average packet rate. Figures 8 and 9 show the variation of packet rates with the level crossings and Weber thresholds respectively. For a typical level crossings threshold of 0.22, the packet rate turns out to be 20 packets/s. For a typical Weber threshold of 0.12, the packet rate turns out to be 23 packets/s. Compared to the haptic loop rate of 1000Hz, the packet rate due to adaptive sampling is much lower.

V. Conclusions

In this paper, we described HoIP - a simple application layer protocol using UDP to network haptic devices. HoIP depends crucially on adaptive samplers and has been designed to give flexibility to the user by allowing different choices of samplers and extrapolators. The code is written in C++ and uses a multi-threaded approach to ensure processing delays are kept to a minimum. In particular, our delay measurement experiments indicate that the worst case processing delay in

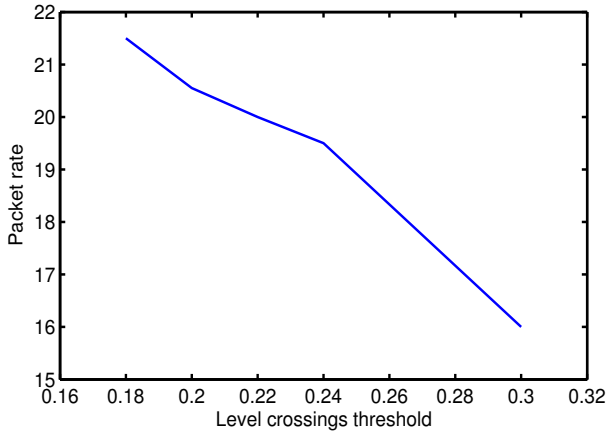

Figure 8 : Graph of packet rate vs level crossings threshold

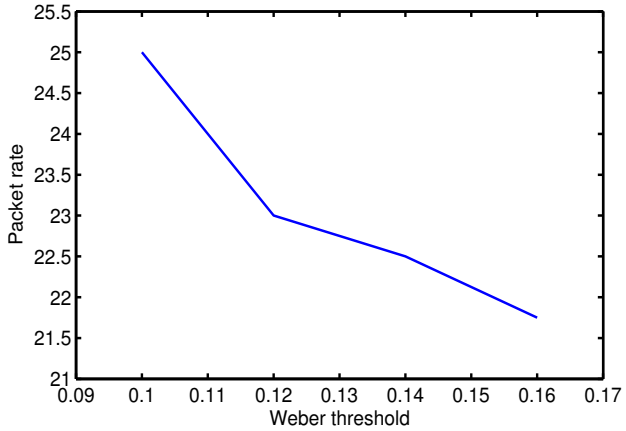

Figure 9 : Graph of packet rate vs Weber threshold

one way communication is less than 0.6 ms. We have also attempted to make HoIP flexible so that it can be made compatible to some of the future developments. HoIP enables researcher and developers to setup a networked haptic environment in order to study haptic signal transmission over real world networks. Towards this end, we aim to make our code available openly to all.

There are several directions for future work. One of the main directions is performance evaluation and tuning of sampling parameters to accommodate non-idealities such as network delay jitter and packet loss. Another direction is development of adaptive algorithms to automatically tune sampling parameters to different users and operating conditions. In a future version, we also aim to integrate audio and video in to HoIP, and add support for multiple users.

REFERENCES

[1] R. Anderson and M. Spong, "Bilateral control of teleoperators with time delay," *Automatic Control, IEEE Transactions on*, vol. 34, no. 5, pp. 494–501, 1989.

[2] B. Chebbi, D. Lazaroff, F. Bogsany, P. Liu, L. Niy, and M. Rossi, "Design and implementation of a collaborative virtual haptic surgical training system," in *Mechatronics and Automation, 2005 IEEE International Conference*, vol. 1, 2005, pp. 315–320 Vol. 1.

[3] K. Sajith, P. Gopal, and S. Chaudhuri, "Hand tremor analysis using deformable object manipulation in a haptic environment," in *Point-of-Care Healthcare Technologies (PHT), 2013 IEEE*, 2013, pp. 13–17.

[4] K. Sreeni, K. Priyadarshini, A. Praseedha, and S. Chaudhuri, "Haptic rendering of cultural heritage objects at different scales," in *Haptics: Perception, Devices, Mobility, and Communication*. Springer, 2012, pp. 505–516.

[5] C. Basdogan, C.-H. Ho, M. A. Srinivasan, and M. Slater, "An experimental study on the role of touch in shared virtual environments," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 7, no. 4, pp. 443–460, 2000.

[6] X. Shen, J. Zhou, A. El Saddik, and N. D. Georganas, "Architecture and evaluation of tele-haptic environments," in *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*. IEEE, 2004, pp. 53–60.

[7] X. Shen, J. Zhou, and N. D. Georganas, "Evaluation patterns of tele-haptics," in *Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on*, 2006, pp. 1542–1545.

[8] K. M. Yap, A. Marshall, W. Yu, G. Dodds, Q. Gu, and R. T. Souayed, "Characterising distributed haptic virtual environment network traffic flows," in *Network Control and Engineering for QoS, Security and Mobility, IV*. Springer, 2007, pp. 297–310.

[9] B. Goode, "Voice over internet protocol (voip)," *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1495–1517, 2002.

[10] S. Clarke, G. Schillhuber, M. Zaeh, and H. Ulbrich, "Telepresence across delayed networks: a combined prediction and compression approach," in *Haptic Audio Visual Environments and their Applications, 2006. HAVE 2006. IEEE International Workshop on*, 2006, pp. 171–175.

[11] O. Dabeer and S. Chaudhuri, "Analysis of an adaptive sampler based on weber's law," *Signal Processing, IEEE Transactions on*, vol. 59, no. 4, pp. 1868–1878, 2011.

[12] P. Hinterseer, S. Hirche, S. Chaudhuri, E. Steinbach, and M. Buss, "Perception-based data reduction and transmission of haptic data in telepresence and teleaction systems," *Signal Processing, IEEE Transactions on*, vol. 56, no. 2, pp. 588–597, 2008.

[13] A. Bhardwaj, O. Dabeer, and S. Chaudhuri, "Can we improve over weber sampling of haptic signals?" in *Information Theory and Applications Workshop (ITA), 2013*, 2013, pp. 1–6.

[14] S. Shirmohammadi and N. D. Georganas, "An end-to-end communication architecture for collaborative virtual environments," *Computer Networks*, vol. 35, no. 2, pp. 351–367, 2001.

[15] S. Paul, K. Sabnani, J.-H. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (rmtp)," *Selected Areas in Communications, IEEE Journal on*, vol. 15, no. 3, pp. 407–421, 1997.

[16] M. Mauve, V. Hilt, C. Kuhmunch, and W. Effelsberg, "Rtp/i-toward a common application level protocol for distributed interactive media," *Multimedia, IEEE Transactions on*, vol. 3, no. 1, pp. 152–161, 2001.

[17] H. Al Osman, M. Eid, R. Iglesias, and A. El Saddik, "Alphan: Application layer protocol for haptic networking," in *Haptic, Audio and Visual Environments and Games, 2007. HAVE 2007. IEEE International Workshop on*. IEEE, 2007, pp. 96–101.

[18] Q. Nasir and E. Khalil, "Perception based adaptive haptic communication protocol (pahcp)," in *Computer Systems and Industrial Informatics (ICCSII), 2012 International Conference on*, 2012, pp. 1–6.

[19] A. Bhardwaj, S. Chaudhuri, and O. Dabeer, "Design and analysis of predictive sampling of haptic signals," *ACM transactions on Applied Perception*, 2013, submitted.

[20] M. Press *et al.*, *Microsoft Win 32 Application Programming Interface: The Programmer's Reference*. Microsoft Pr, 1992, vol. 1.

[21] "Hapi reference: www.h3dapi.org," july 2012.

[22] P. Kadlecek, "Overview of current developments in haptic apis," *Proceedings of CESCG*, 2011.

[23] "Phantom omni device reference: www.sensable.com/haptic-phantom-omni.htm," july 2012.

[24] A. Silva, O. Ramirez, V. Vega, and J. Oliver, "Phantom omni haptic device: Kinematic and manipulability," in *Electronics, Robotics and Automotive Mechanics Conference, 2009. CERMA '09.*, 2009, pp. 193–198.