Admux: An Adaptive Multiplexer for Haptic–Audio–Visual Data Communication

Mohamad Eid, Jongeun Cha, and Abdulmotaleb El Saddik, *Fellow, IEEE*

***Abstract*—Research trends in multimedia strive to incorporate multiple modalities, such as audio, video, graphics, and haptics, into multimedia applications to enhance the user's experience. Researchers have made significant progress in advanced multimedia by incorporating virtual reality environments, haptics, and scent into the human computer interaction paradigm. However, the communication of multimedia data over the Internet (particularly the haptic media) remains a real challenge since each media has varying and sometimes conflicting communication requirements. This paper proposes Admux, an adaptive application layer multiplexing framework (including a communication protocol) for multimedia applications incorporating haptic, visual, auditory, and scent data for nondedicated networks. Being an application layer framework, Admux is highly adaptable to the application requirements, the media type (haptic, audio, video, etc.), and the network conditions. To facilitate the application–Admux communication, we used haptic application metalanguage descriptions. Second, Admux enhances the network throughput by adopting statistical multiplexing. Finally, Admux enables media prioritization based on the application events and QoS requirements. By simulating an interpersonal teleconferencing system (named the HugMe system), our results showed that Admux provides dynamic bandwidth allocation based on the network conditions, media type, and application events.**

***Index Terms*—Adaptive communication framework, audio–visual communication protocols, haptics, statistical multiplexing, telehaptics.**

I. Introduction

WHEN two humans communicate in a face-to-face fashion, they do so by exchanging information via several media cues. Common communication signals include visual cues (such as gaze, facial expressions, and gesture), auditory cues (such as intonation and voice quality), and haptic cues (such as shoulder tip, handshaking, and tangents). These modalities play a prominent complimentary role in both the semantic and the socioemotional aspects of communication [1]. For example, a formal spoken statement accompanied by a particular facial expression (visual) and a shoulder tip (haptic) might be interpreted as a joke. Therefore, face-to-face interaction is naturally a form of multimodal communication.

Manuscript received December 14, 2009; revised March 10, 2010; accepted April 1, 2010. Date of current version December 8, 2010. The Associate Editor coordinating the review process for this paper was Dr. Augusto Sarti.

The authors are with the MCRLab, University of Ottawa, Ottawa, ON K1N 6N5, Canada (corresponding author Mohamad Eid e-mail: eid@mcrlab.uottawa.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2010.2065530

A. Haptic–Audio–Visual Multimedia Applications

Augmenting a real scene with virtual objects has recently received significant attention; a research field referred to as mixed reality [2]. Adding virtual cues enhances the quality of user experience in several applications ranging from entertainment and gaming to medical training and military applications. Similar applications can be found in [3] and [4].

Haptics plays a prominent role in making virtual objects physically palpable in a collaborative and/or networked virtual environment. For instance, the learning tool proposed in [5] allows users to learn handwriting by means of force feedback to handwrite characters from Japanese, Arabic, English, French, and Spanish languages. The authors in [6] reviewed the applications of haptics in various multimedia applications and systems. Therefore, a haptic modality has been proven effective in applications of learning physical skills.

There have been few efforts to add the sense of smell or scent to the universe of multimedia. There is already a commercial device called the Scent Dome—developed and marketed by Trisenx—that allows users to smell products sold online or create and e-mail their own scents [7]. Indeed, the sense of smell is envisioned to be crucial media in multimedia systems in the few years to come.

Therefore, the trend with multimedia applications and systems is the incorporation of multiple media in order to enhance the user's quality of experience (QoE). The relationships between the different media, how they interact with each other and how a human perceives them, have been studied by many researchers [8]. On the other hand, the communication of such multimedia information over the Internet remains a challenging research question.

B. Haptic–Audio–Visual Communication Requirements

The networking requirements are defined using the quality of service (QoS) model that quantizes the quality of information delivery as a set of quality parameters that can be at both the application and networking levels. In the networking terminology, QoS aims at increasing the overall utility of the network by granting priority to higher-value or more performance-sensitive flows [9]. The authors in [9] claim that four QoS parameters are the most crucial for teleconferencing applications: delay, jitter (the variation of the network delay), data loss rate, and data rate [9].

Video conferencing applications are delay sensitive and loss sensitive by nature [9]. The QoS requirements for video

TABLE I
QoS PARAMETERS FOR FOUR MEDIA: AUDIO, VIDEO, GRAPHICS, AND HAPTICS

QoS Parameter	Audio	Video	Graphics	Haptics
Delay	≤ 150 ms	≤ 400 ms	[100-300] ms	[3-60] ms
Jitter	≤ 30 ms	≤ 30 ms	≤ 30 ms	[1-10] ms
Data Loss Rate	≤ 1%	≤ 1%	≤ 10%	[0.01-10]%
Data Rate	[22 Kbps – 200 Kbps]	[2.5 Mbps –40Mbps]	[45 Kbps – 1.2 Mbps]	[128 Kbps]

communication are specified in [10] as follows: 1) Transfer delays should always be less than or equal to 400 ms to ensure an acceptable quality of a perception level; 2) the jitter effect should be at a maximum of 30 ms; 3) the bandwidth requirements, compared to other media, are the most demanding (from 2.5 to 5 Mb/s); and 4) the data loss rate should always be within 1% for stable video rendering.

As for audition, the delay constraints can be either tight or relaxed—depending on whether the application is interactive or not. However, as a rule of thumb, humans expect higher fidelity audio and do not usually tolerate quality degradations [9]. Bandwidth is not a major concern for audio traffic; typical values range from 22 to 200 kb/s. The critical audio requirements are the end-to-end delay, jitter, and loss. Typical delays in audio data transfer are bounded to 150 ms, whereas jitter should be maintained at a maximum of 30 ms. As in video, the audio data loss rate should be less than 1%.

The quality of a virtual/graphics cue is commonly evaluated at the application and user levels and is usually referred to as QoE [11]—such as user preferences, user performance, satisfaction, etc. However, the network QoS requirements have a direct impact on the overall QoE [12]. Several studies were conducted to derive the network QoS requirements for graphics and virtual object communication, and the findings were as follows: transfer delay ranging from 100 to 300 ms, jitter less than or equal to 30 ms, a packet loss rate of 10% or less, and a bandwidth ranging from 45 kb/s to a maximum of 1.2 Mb/s [12].

The concept of sending haptic data over a network is referred to as "telehaptics" [13]. One unique requirement in telehaptics is that unwanted vibrations and unbounded forces are not only distracting but also potentially unsafe for the human operator. Many network impairments, such as delay, jitter, and packet loss, can easily result in unstable haptic rendering, and therefore, haptic data communication is the most demanding as for the QoS requirements [10]. The latency requirement is very strict for haptic data, is largely affected by the performed task (tactile or kinesthetic feedback), and usually ranges between 3 and 100 ms [14], [15]. It has also been shown that jitter has the greatest impact on the coordination performance when the latency was high and the task was difficult (1 to 10 ms) [16]. The bandwidth requirement is more relaxed than the delay and jitter requirements, which is around 128 kb/s [10]. Finally, the data loss rate ranges between 0.01% and 10% [10].

Table I summarizes the QoS parameters associated with the four media: audio, video, graphics, and haptics. Notice that these requirements are very varying per media as well as among the multiple media. It is also clear that haptic media is more sensitive to delays and jitter than other media types. Therefore, in this paper, we propose Admux. Adapting to the different and varying multimedia requirements is the challenge we are investigating in this paper.

II. RELATED WORK

The problem of communicating multimedia data, particularly haptic media, has been the subject of research for the last decade. There have been three research approaches: 1) Improve the control mechanisms to accommodate the unpredictable behavior of the Internet (such as the use of delay compensation techniques [17] and jitter smoothing algorithms [18]); 2) improve the QoS of the Internet, and make it as reliable as a dedicated network; and 3) optimize the volume of communicated data necessary to maintain the overall quality of user perception.

A. Compression and Control

The compression research for audio, video, and graphic data has reached a matured stage (realized in public standards such as MPEG, H.261, H.262, etc.). However, despite the stringent need for haptic data compression, the field is still in its infancy. There are three approaches for haptic data compression: 1) developing real-time compression for heterogeneous haptic information [19]; 2) exploring the suitability of existing compression techniques for haptic data [20]; and 3) minimizing the amount of data using perception-based data reduction [21]–[23].

On the other hand, control techniques are used to ensure the consistency for haptic applications such as dead reckoning (DR). In DR, each object's state is updated locally, and if the difference between the current state and the previous state is larger than a preset threshold, an update is broadcasted to all participants. The main difficulty with this algorithm is its potential to produce short-term inconsistencies during the transmission of an update destined to correct the error in the predicted state. Mauve *et al.* have proposed an alternative approach to solve the problem of inconsistency using the local lag and the timewrap algorithms [24].

Another common control technique is called local lag [24], where it is assumed that each operation is qualified with two time stamps: the time when the operation is issued and the time when the operation should be executed at all nodes concurrently. The difference between these two time stamps is called the lag value [25]. The variation of the network delay is known as jitter. One way to smooth the jitter has been proposed by Gautier *et al.* [26] where messages are multicast to all participants who share a synchronized clock. A time stamp is attached to each update message, and at the receiver side, the processing of the received packet depends on their time stamps and not on their time of arrival.

Admux, on the other hand, addresses different problems other than the local lag or jitter smoothing; in fact, Admux can be used together with these algorithms to ensure haptic media stability and to optimize fidelity. On the other hand, Admux

dynamically distributes the network resources to best match both network and application requirements.

B. Transport and Network Protocols

The generic transport-layer protocols [namely the transmission control protocol (TCP) and user datagram protocol (UDP)] are used by several multimedia applications. Few protocols have been proposed and evaluated for haptic applications (such as synchronous collaboration transport protocol (SCTP), smoothed SCTP, light TCP, real-time protocol for interactive applications (RTP/I), and transport for teleoperations over overlay networks).

The TCP provides several services that have a negative impact on haptic applications. If a haptic update is lost during transmission, all the updates that follow will be buffered on the receiver side, and thus, the fresh updates are being needlessly buffered on the receiver side [27]. The UDP does not suffer from such drawbacks; however, it does not fully suit the reliability requirements of multimedia applications [27]. For instance, the UDP has no buffering delays and suffers from higher jitter because it does not resend lost packets [27].

The SCTP is similar to the UDP since most of the messages are sent unreliably, whereas key messages are sent reliably. The smoothed SCTP adds a jitter smoothing mechanism to the SCTP. On the receiver side, each received update is placed in a bucket according to its time stamp. The receiver constantly checks for updates that have been sent by each δt (milliseconds), and in case an update is found, it is retrieved from the bucket and forwarded to the application [28].

The light TCP is inspired from the TCP and supports the concept of message obsolescence. The sender queue accepts update messages from the application and processes them as follows [29]: 1) A key update is placed at the end of the queue and marked as a key message; 2) a normal update can replace older normal updates; and 3) unacknowledged update messages are placed again in the queue if no newer updates from the same object have been produced.

The RTP/I is an application layer protocol. The communicated messages are categorized as follows: event, state, and request-for-event packets [30]. An event packet carries an event or a fraction of an event. A state packet carries the entire state of an object in the environment. The request-for-event packets are used by participants to indicate that the transmission of an object's state is required by the sender [30].

The research presented in [31] proposes a framework of the QoS management for supermedia teleoperation systems, where latency-sensitive supermedia streams are encoded using redundancy codecs and transmitted over multiple overlay paths. The overlay routes and encoding redundancy can be dynamically tuned to meet the QoS requirements of the supermedia streams to compensate for the network performance degradation.

Admux is an application layer protocol and, thus, has the ability to adapt to application interactions and events. Admux utilizes a description language [haptic application metalanguage (HAML)] that enables applications to configure the Admux communication protocol based on the application needs and requirements.

C. Statistical Multiplexing Schemes

Statistical multiplexing is a proven technique used to improve the efficiency of communication over a limited bandwidth network [32]. The principle is that a group of media channels shares a limited quantity of bandwidth. The bandwidth is allocated on a frame-by-frame basis by a centralized controller (the multiplexer) so that the channels with the most demanding QoS requirements are allowed to borrow more bandwidth than the channels with less QoS requirements. Most statistical multiplexing research is done at the transport level (particularly asynchronous transfer mode networks [32]).

One related work is to dynamically control the arrival rate of multimedia data by switching the coders to different compression ratios based on the network conditions [33]. The technique is tested with audio media, and it was found that the link performance has significantly improved in terms of reducing the probability of call blocking and enhancing the multiplexer gain. The work in [34] investigated the use of self-organizing neural networks to design a statistical multiplexer for video streams. The proposed approach uses multiple video coders, followed by a multiplexer that generates the aggregate sequence for a few video streams. The neural network approach performed very well to improve the packet loss as compared to round-robin approach [35] and to smooth the variations of delays with the rate control.

Statistical multiplexing is realized at the networking level. Admux, on the other hand, implements a statistical multiplexing scheme at the application layer. This implies that Admux multiplexing is adaptive to application level interactions and events, thus providing higher controllability for the communication requirements.

D. Multimedia Rate Shaping and Synchronization

Rate shaping techniques attempt to adjust the rate of multimedia traffic generated by the encoders according to the current network conditions. Feedback mechanisms are needed to detect changes in the network and adapt the rate of the encoders. The rate shaping can be obtained by changing the media frame rate, quantization settings, and movement detection threshold.

Adaptive synchronization can be used for multimedia teleconferencing systems involving multiple channels of communication [36]. The sender is assumed to put a time stamp in the packets. At the receiver, the playback clock (PBC) and three counters for no-wait, wait, and discard packets are maintained. The algorithm specifies that, when packets arrive early and enough wait packets have been received, the PBC is incremented. Similarly, when the thresholds of no-wait/discard packets are received, the PBC is decremented.

Multimedia rate shaping and synchronization have been thoroughly investigated for audio, video, and graphic media. However, dealing with haptic data is yet to be investigated. Admux explores the possibility of combining and synchronizing haptic media with audio and video information.

In this paper, we propose Admux, a communication framework with three distinguished features: adaptability and media prioritization, synchronization of haptic with audio/video streams, and statistical multiplexing. First, Admux adapts to

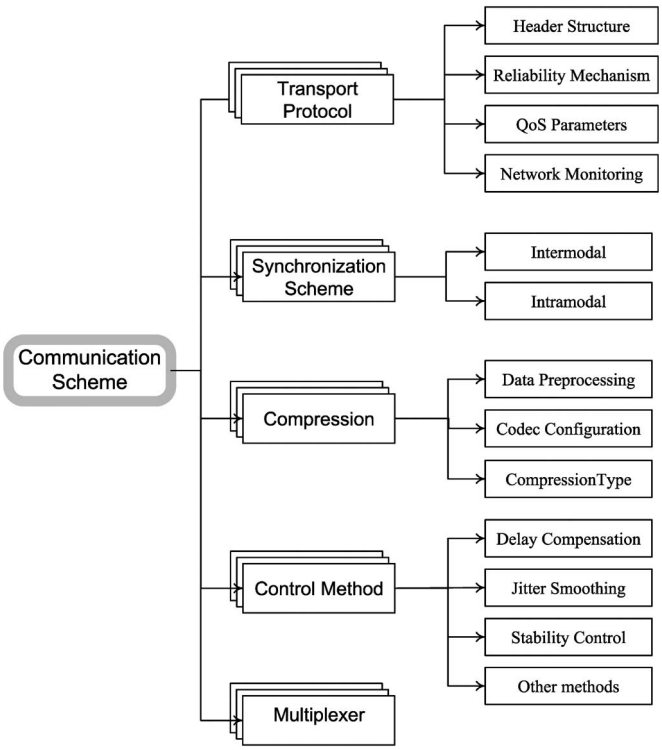

Fig. 1. Communication scheme.

```
<?xml version="1.0" ?>
<CommunicationDS xmlns=http://www.mcrlab.uottawa.ca
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="... ">
   <TransportProtocol>
    <HeaderStructure>UDP</HeaderStructure>
    <Reliability>Negative Acknowledgment</Reliability>
    <QoSParameters>
      <Delay>50</Delay>
      <Jitter>10</Jitter>
      <DataRate>10M</DataRate>
      <LossRate>10^10</LossRate>
    </QoSParameters>
    <MonitoringProtocol>ICMP</MonitoringProtocol>
   </TransportProtocol>
  <Synchronization>
    <Intermodal>TimeStamp</Intermodal>
    <Intramodal>Sequence Number</Intramodal>
    </Synchronization>
  <Compression>
    <Video> ... </Video>
    <Haptic>... </Haptic>
  </Compression>
   <ControlMethods> ...</ControlMethods>
   <Multiplexer> ...</Multiplexer>
</CommunicationDS>
```

Fig. 2. Excerpt of the communication scheme.

the application requirements, the media type, and the network resources. Second, Admux provides a synchronization scheme for haptic–audio–video data. Finally, Admux intelligently multiplexes the input media channels into a continuous stream to realize efficient dynamic bandwidth allocation. The application simulations confirmed our research proposal in terms of adaptability to network conditions, application requirements, and media type. However, the QoS parameters are not absolutely guaranteed (statistical multiplexing). Thus, delay compensation and jitter smoothing are still needed to guarantee haptic interaction stability.

III. Admux Multiplexing Framework

This section introduces the communication framework where the multiplexing scheme will be used. The core part of this section is the mathematical model for the adaptive multiplexer. We present the model and provide insights to its comprising factors/coefficients.

A. HAML

The HAML is designed to provide a technology-neutral description of haptic models. It contains ergonomic requirements and specifications for haptic hardware and software interactions where haptic application components such as haptic devices, haptic application programming interface, or graphic models make themselves and their capabilities known. A complete description of the HAML can be found in [38].

One crucial part of the HAML is the communication description scheme, shown in Fig. 1. The communication scheme describes the QoS requirements for each media channel. The scheme comprises five different categories of communication specific information: the transport protocol, synchronization scheme, compression, control method, and multiplexer (as shown in Fig. 1). The transport protocol defines the transport-layer protocol, the reliability mechanism, the QoS parameters describing the current network resources, and the network monitoring protocol used to measure the current network QoS (such as the Internet control message protocol). The synchronization scheme defines both the intermodal and intramodal synchronization models used in the communication. The compression scheme defines the data preprocessing method (if any is used) and the codec configurations. The control method scheme describes the algorithms that are used at both ends of the network to compensate for the network deficiencies. Fig. 2 shows an excerpt of the HAML communication scheme.

B. Communication Framework

The general overview of the Admux communication framework is described in our previous work [37]. The communication framework consists of the following building blocks.

1) *Multiplexer/Demultiplexer:* This component provides a way to interleave data from different input media channels into one serialized bitstream.
2) *Network Interface:* The network interface provides the transport-layer communication mechanism for multimedia communication (using the UDP in our case).
3) *Channel Codec:* This component defines two types of coders/decoders (codecs) for both control channels and media channels.
4) *HAML QoS:* This component implements the HAML schema to define the QoS parameters per each input channel. A comprehensive description of the HAML is presented in [38].

C. Mathematical Modeling

1) Definitions: Given N input channels represented by two vectors: the desired selection probability (DSP) vector (X) that is computed from the desired QoS requirements for each input channel [shown in (1)] and the current selection probability (CSP) vector (W) that represents the currently allocated resources for each channel and is shown in (2). The objective is to match the two vectors so that each channel is provided the desired proportion of the network resources

$$X = [x_1, x_2, \ldots, x_N]^T \tag{1}$$

$$W = [w_1, w_2, \ldots, w_N]^T. \tag{2}$$

The DSP vector represents the static requirements of the application; its values do not change over the multiplexing iterations. It is calculated as the normalized weighted average of the desired QoS parameters, as shown in (3) (M is the number of QoS parameters). The multiplexer is statistical since it uses probabilities that are mapped to network resources. For instance, the higher the selection probability, the better the chance of a channel to win a resource, and thus, the channel is statistically given a higher portion of network resources.

On the other hand, the CSP vector is dynamic and is computed according to (4). The initial value of the CSP vector (at $t = 0$) is computed as the normalized fractions of the initial network resources and is in proportion to the DSP values. At a general instance of time $(t > 0)$, the CSP is updated based on two factors [shown in (4), $t > 0$]. First, the weighted difference $(x_i - w_i(t))$ is added for the winning channel and subtracted from all other channels since it represents a resource that is given to the winning channel and taken from others. A smaller difference $(x_i - w_i(t))$ results in slower convergence of the CSP value to its DSP counterpart. This is desired since the other channels should be provided chances to win the competition next time and thus provide fair distribution of resources. Any changes in the network conditions are incorporated using the $\Delta Q(x_i / \sum_{k=1}^{N} x_k)$ term. The updates in the network resources are distributed proportional to the DSP weights

$$x_i = \sum_{j=1}^{M} \alpha_j \cdot A_{p_j} \tag{3}$$

$$w_i(t) \begin{cases} \dfrac{x_i \sum_{k=1}^{M} \beta_k \cdot N_{p_k}}{\sum_{l=1}^{N} x_l} & t = 0 \\ w_i(t-1) + a_i \left(x_i - w_i(t)\right) Z_i & \\ \quad + \Delta Q \dfrac{x_i}{\sum_{k=1}^{N} x_k} & t > 0 \end{cases} \tag{4}$$

where

$$Z_j = \begin{cases} 1, & \text{for the winner} \\ -\frac{1}{N-1}, & \text{otherwise} \end{cases}$$

M number of QoS parameters;
N number of input channels;
x_i DSP of the ith channel;
w_i CSP based on the network conditions;
α_{ji} convergence rate of the ith input QoS parameter;
A_{pi} desired value of the ith input QoS parameter;
β_k weighting factor for the kth network parameter;
N_{p_k} value assigned to the kth network parameter;
a_i rate at which w_i converges to the x_i value;
ΔQ change in the network conditions.

Additionally, the proposed model defines an absolute minimum DSP vector $(X_{(\min)})$ that represents the nontolerable minimum QoS requirements corresponding to the just perceivable quality of the media channels [see (5)]. If the minimum requirement of any channel is violated, the multiplexer drops that channel where the minimum QoS requirements cannot be guaranteed and redistribute its resources to the other channels. In this case, the performance of Admux will be the same except for $N - 1$ rather than N channels

$$\mathbf{X}_{(\min)} = \left[\mathbf{x_1}_{(\min)}, \mathbf{x_2}_{(\min)}, \ldots, \mathbf{x}_{N_{(\min)}}\right]^T. \tag{5}$$

On the other hand, the expected output is the selection sequence of input channels as well as the prioritization vector that enforces media prioritization. The prioritization factor can be used to derive the compression rates in order to control the data rate, error rate, and jitter effects.

2) Procedure: The procedure used to determine the winner input and eventually make a selection is composed of three steps: Compute the prioritization factor, compute the intensity of each input and make a selection according to the minimum intensity, and update the CSP values.

1) Step 1: The prioritization factor (C_i) is computed as defined in (6). C_i is the summation of $1/N$ (the equal share of weights), the contribution weight $[f_i/(k)]$, and the buffer weight $(q_{l_i} \cdot \varphi_i)$. The contribution weight $f_i(k)$ represents the history of winnings for a channel (K_d) and enables controlling the prioritization factor in an exponential manner using the exponent prioritization coefficient (q_{e_i}), as shown in (7). The buffer weight is computed, according to (8), as the summation of the number of data units in the buffer (P_{size_i}) multiplied by the weight coefficient (ρ_i) and the rate of data unit flow (P_{rate_i}) multiplied by its respective weight coefficient (σ_i). The linear prioritization coefficient (q_{l_i}) enables controlling the prioritization factor in a linear manner (which may be desired for some media channels)

$$C_i = g\left(\frac{1}{N} + f_i(k) + q_{l_i} \cdot \varphi_i\right) \tag{6}$$

$$f_i(k) = 1 - \left(1 - \frac{1}{N}\right)^{\frac{k_{j_i}}{K_{d_i} q_{e_i}}} \tag{7}$$

$$\varphi = \rho_i P_{\text{size}_i} + \sigma_i P_{\text{rate}_i} \tag{8}$$

where

g fairness gain constant empirically set to ten;
$f_i(k)$ contribution of the ith input channel;
k_i transport stream packet (TS packet) size for the winner channel;
K_{d_i} number of wins for the ith channel;
q_{e_i} exponential prioritization coefficient;
φ_i state of the transmit buffer of the ith channel;
q_{l_i} linear prioritization coefficient.

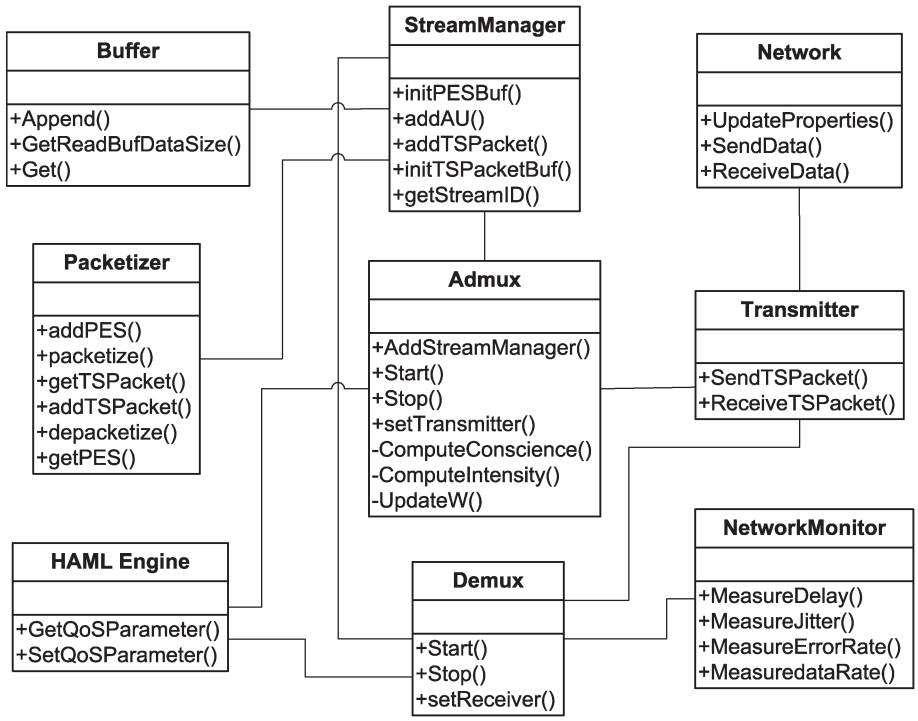

Fig. 3. UML class diagram for Admux simulation.

2) Step 2: Calculate the intensity $I_i(t)$ for each input channel (which represents the distance between w_i and x_i minus the prioritization factor), as shown in (9). The channel with the minimal intensity wins the competition and, thus, is selected by the multiplexer. Notice that $D(w_i(t), x_i(t))$ is a negative value since w_i is always smaller than x_i. Therefore, the higher the $C_i(t)$, the smaller the intensity becomes (larger negative number), and thus, the more likely the channel is to be selected

$$I_i(t) = D\left(w_i(t), x_i(t)\right) - C_i(t) \tag{9}$$

where

$D(w_i, x_i)$ Euclidean distance between w_i and x_i;
$C_i(t)$ prioritization factor, which is used to enforce media prioritization.

3) Step 3: Read the current network conditions and update the CSP values for all the input channels according to (4) $(t > 0)$. Notice that the w_i weight will increase for the winner channel, whereas the w_i weights for all other channels will decrease (since a resource is taken away from all other channels and given to the winner channel). Additionally, the convergence rate (a_i) controls how fast the CSP values converge to the DSP values.

D. Admux Implementation

The multiplexer simulation was developed in C++ using the Microsoft Visual Studio 2005. The implementation library consists of four primary classes and three secondary/helping classes. Fig. 3 shows the major classes used in the Admux framework implementation. The primary classes are `StreamManager`, `Admux`, `Network`, and `Demux`. The other three classes, namely `Buffer`, `Packetizer`, and `Transmitter`, play secondary roles in the simulation. Here is a brief explanation of the classes and how they interact with each other.

1) `StreamManager`: This class is responsible for managing the input channels to be multiplexed. It can create, update, and retrieve information about a particular channel in the communication system so that every input channel is mapped into one instance of this class. Furthermore, the `StreamManager` instantiates one or more instances of the `Buffer` class to be attached to each channel—depending on whether the channel has one or more buffer. Finally, the `StreamManager` class provides the `Admux` class access to the channel's information, the state, and the data of the buffers.
2) `Admux`: This class is the core class of the simulation. It implements the multiplexing model proposed in the previous section. The multiplexing is performed by calling several private methods in the order [`ComputeIntensity()`, `ComputePrioritization()`, and `UpdateW()`]. In addition, the class provides access to the application to start and/or stop the multiplexer using the two public methods: `Start()` and `Stop()`. Finally, the application can add channels to the multiplexer using the method `AddStreamManager()`.
3) `Buffer`: This class manages the buffer associated with a particular communication channel. With this class, we can append data to the buffer (using the `Append()` method), retrieve the buffer size (using the `GetRead BufDataSize()` method), and retrieve the buffer information (using the `Get()` method).
4) `Packetizer`: This class is responsible for segmenting the received data into a constant size data unit before storing it in the buffer. A frame of the "raw" media data is referred in the implementation as a packetized elementary

stream packet (PES packet), which might have a variable length. Therefore, the PES packets are fragmented into smaller fixed sized data units called a TS packet. This ensures that the data rate of the multiplexed stream is constant. The `packetize()` method receives an instance of a PES packet and breaks it down to a number of TS packets. The `depacketize()` method does exactly the opposite. It receives a number of TS packets and assembles them into one PES packet and then forward the PES packet to the receive buffer.

5) `Transmitter`: This class provides the `Admux` class the interface to the transport protocol. It handles communicating the TS packets into the network using (in the current implementation) the UDP as a transport protocol (using the `SendTSPacket()` method). It also retrieves data from the network and forwards them to the `Demux` class at the receiver side (using the `ReceiveTSPacket()` method).
6) `Demux`: This class implements the demultiplexing of the multimodal stream into the destination channels. The received data are passed to the `Demux` class via the `Transmitter` class and demultiplexed and forwarded to the corresponding stream manager (to be depacketized into PES packets and stored in the receive buffer). The application controls the behavior of this class using the two methods `Start()` and `Stop()`.
7) `Network`: This class simulates the network conditions. The simulated QoS parameters are the network delay, jitter, data rate, and data error rate. The `Transmitter` class sends the data to the `Network` class (using the `SendData()` method), which in turn applies the appropriate network conditions and returns the data using the `ReceiveData()` method. The network properties can also be updated using the `UpdateProperties()` method.
8) `HAML Engine`: This class enables the multiplexer to read the desired QoS parameters from the HAML file using the `GetQoSParameter()` method. Furthermore, the multiplexer can update the list of the desired QoS parameters using the `SetQoSParameter()` method.
9) `NetworkMonitor`: This class measures the network QoS parameters for each communication channel. In the current implementation, this class implements computing the average delays, jitter, data rates, and data error rates for the four media channels. This class is used only for measurements and simulation analysis. It has four methods, namely `MeasureDelays()`, `MeasureJitter()`, `measure ErrorRate()`, and `MeasureDatarate()`, each returning an array of values for the respective quality parameters. For instance, `MeasureDelays()` returns an array containing the average delays for the four media channels as computed at the demultiplexer side.

E. Admux Stability Analysis

As a communication framework, it is important to assess the stability of the Admux framework. The approach adopted in our analysis (proposed in [40]) identifies two categories of evolution characteristics, namely architectural and individual class metrics. In our analysis, we refer to hypothesis 2, which states that a stable framework has a number of single inheritance instances (NSI) to design size in class (DSC) ratio just above 0.8 if multiple inheritances are seldom used—which is our case. Our results show that the DSC is 24, whereas the NSI is 21, and thus, the NSI/DSC ratio is 0.875. This shows that the Admux architecture is thus stable.

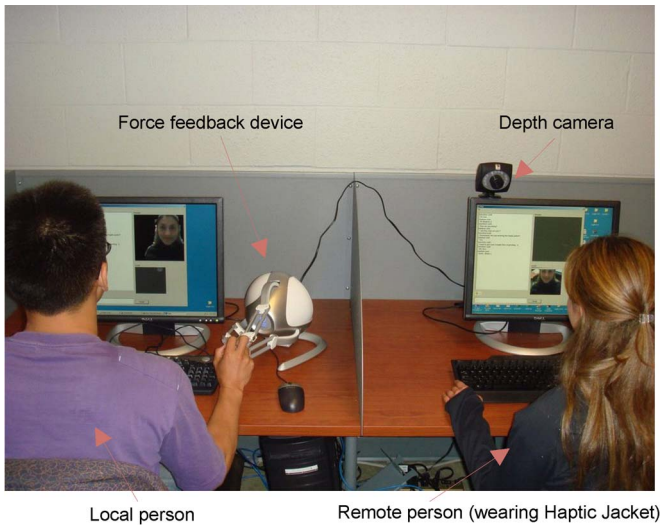

Fig. 4. HugMe system with local and remote users.

IV. Application Simulation and Results

A. Application Scenario

Fig. 4 shows an interpersonal communication application, named the HugMe system, which we have simulated as a proof-of-concept application for the Admux communication framework. As shown in Fig. 4, the remote person is wearing a haptic suit (haptic jacket) that is capable of simulating the nurture touching. The local person uses a haptic device to communicate his feelings with the remote person. A depth camera [39] is used to capture the image and depth information of the remote person and send it back. The local person can touch the video contents, whereas the remote person receives synchronous touch via the haptic jacket.

The four simulation streams are defined as follows: The video object is simulated as a 30-frames/s stream of video frames, each with a size of 640 $\times$ 480 pixels (with 32 b/pixel (24 b for a red, green, and blue image and 8 b for the depth image), we assume a frame size of approximately 10 kB). The haptic object is simulated with an update rate of 1-kHz frame rate with a haptic frame size of 32 bytes (position, 12 bytes; orientation, 12 bytes; and time stamp, 8 bytes). The audio object is simulated at a 64-kb/s rate with a frame size of 417 bytes (stereophonic). The graphic object is kept at 30 frames/s, and the frame size is assumed to be 100 bytes (the graphic frame is the update message for the position and/or orientation of the graphic objects). The simulation traffic is composed of 1000 packets per input media channel, and all the results are averaged over the 1000 samples. Finally, a Pentium IV PC with 2 GB of RAM and 100-Mb/s Ethernet cards was used for the simulation.

TABLE II
COMPARING DELAYS/JITTERS FOR PARALLEL AND MULTIPLEXED CHANNELS (NETWORK DELAY IS 30 ms AND JITTER IS 10 ms)

	Approach	Haptic	Audio	Graphics	Video
Frame size (bytes)		32	417	100	10K
Frame rate (fps)		1000	8000	30	30
Delay (ms)	Multiplexer	35.34	70.55	57.34	252.54
	Parallel	103.9	104.9	102.88	105.14
	Desirable	30	100	60	300
Jitter (ms)	Multiplexer	7.98	26.53	33.98	44.62
	Parallel	28.55	27.28	29.12	28.19
	Desirable	10	30	40	50

B. Simulation Criteria and Results

Six distinguished features of Admux are evaluated: 1) the comparison between the Admux multiplexing scheme and the parallel communication approach; 2) the adaptability of multiplexing to changes in the network conditions; 3) the time complexity; 4) the adaptability to the application events and/or interactions and multiple-buffering scheme; 5) the multiple-buffering scheme; and 6) the Admux scalability.

1) Multiplexing Scheme Versus Parallel Communication: The objective of this test is to compare the performance of Admux with the approach of sending four parallel streams. To this end, we have simulated four channels with the same QoS parameters using four UDP communication channels. The comparison is made for the delay and jitter attributes. In both cases, the network delay is set to 30 ms while the jitter is set to 10 ms. A random number generation function is used to generate random numbers between 20 and 40 ms.

As shown in Table II, the average delays and jitter are almost equal in the case of the parallel channels. This is because the application cannot enforce any prioritization for the QoS requirements. Because of the dynamic prioritization mechanism, Admux provided each channel with proportional resources based on their QoS and media requirements. The Admux prioritization mechanism works as follows: As C_i increases, the intensity of the input decreases which gives the corresponding channel a better chance to win the competition. Moreover, notice that C_i changes linearly with the state of the input buffers and exponentially with the contribution factor $(f_i(k))$. The two coefficients q_{l_i} and q_{e_i} can be adjusted at run time to dynamically change the prioritization weights of the media channels.

As shown in Table II, the delay and jitter requirements are met for all the channels—except with the delay for the haptic media which is due to limited resources—(unlike the parallel communication approach where these requirements were not fairly distributed). This shows that Admux can adapt to the application QoS requirements, particularly when they are different for each media channel.

2) Adaptability to Network Conditions: This test evaluates the ability of the multiplexer to adapt to changes in the network conditions (resources). This can be demonstrated by tracing how the weighting factors for each channel are changing against the desired weights. Due to the fact that the weighting factors represent the current prioritization level for each media channel, Fig. 5 shows how the weighting factors (w_i) are converging to the desired values (x_i) over time. Notice that the optimal performance is achieved when the weighting factors are equal to the desired values, but due to limited network resources, this cannot always be achieved. Therefore, the resources are redistributed based on the priority level of the respective media and network conditions.

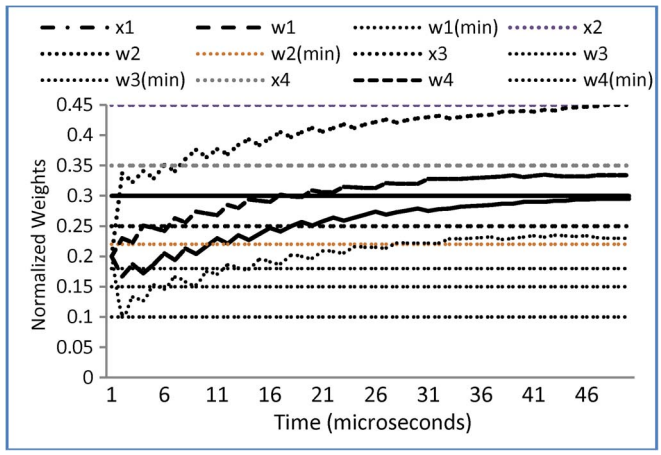

Fig. 5. Simulation with four channels (adaptability of the multiplexer).

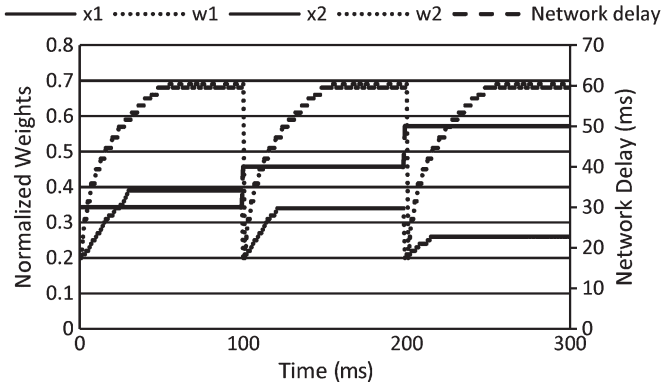

Fig. 6. Simulation of two streams [haptic $(x1, w1)$ and video $(x2, w2)$] with three network conditions (delay = 30 and jitter = 5, delay = 40 and jitter = 10, and delay = 50 and jitter = 15).

Another simulation we performed was to evaluate the adaptability of Admux to changes in the network conditions. This is performed by tracing how the w_i values can converge to x_i in proportion to their corresponding QoS requirements. Fig. 6 shows the simulation with haptics ($x1$, $w1$—high priority) and video ($x2$, $w2$—low priority) with three network conditions (delay $= 30$ and jitter $= 5$, delay $= 40$ and jitter $= 10$, and delay $= 50$ and jitter $= 15$). As shown in Fig. 6, $w1$ is converting to $x1$, whereas $w2$ is deviating from $x2$; this implies that network resources are taken from the video stream (higher $x2 - w2$) and given to the haptic stream ($x1 - w1$ almost zero). Therefore, changes in the network conditions would affect some streams where higher priority streams (such as haptic media) can maintain enough resources. This, in our opinion, is the main contribution of Admux.

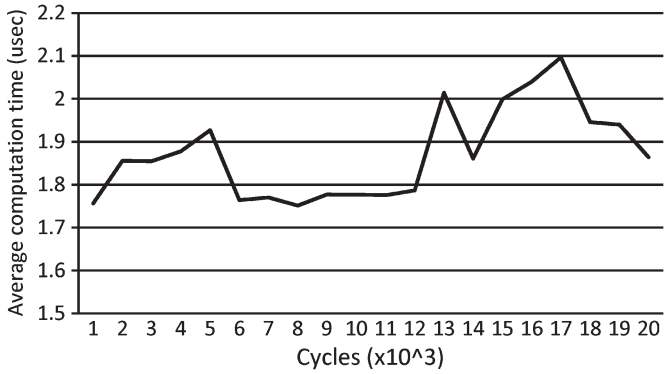

Fig. 7. Time complexity analysis.

3) Time Complexity Analysis: One of the critical factors to examine is the time complexity of the multiplexing scheme. We implemented a high precision timer (up to nanoseconds resolution), which is provided in a Windows XP operating system, in the `NetworkManager` class to measure the computation time for a multiplexing cycle. As the results in Fig. 7 demonstrate, the average computation time is 1.87 μs (the maximum computing time is 2.1 μs which is comfortably below the 1-ms delay needed for the haptic modality). This implies that the multiplexing computation time has no tangible impact on the Admux performance.

4) Application Event Adaptability: The objective of this simulation is to evaluate the adaptability of Admux to application events and interactions. We simulated a collision event between the haptic device and the body of the remote person. Typically, the audio media is assigned a higher priority level than the video; however, we assume that the video stream should be given a higher priority level than the audio in case of collision (the same for the haptic media). Admux should provide a higher bandwidth proportion to the video stream compared to the audio stream as soon as the collision event occurs. We measure the dynamic bandwidth allocation based on the number of packet sent for each media channel.

The simulation results are shown in Fig. 8 where the number of communicated packets is plotted against time in two cases: when the collision event was considered by Admux versus when it was ignored. The two streams are noted as V object (for the video stream) and A object (for the audio stream) in the diagram. In case the collision event is not considered, the A object received a higher bandwidth than the V object since, in general, the audio channel has a higher priority than the video channel. On the other hand, when the collision event is simulated at $t = 40$ s (see Fig. 8), Admux dynamically changed the prioritization of the two media channels so that the V object is allocated a higher bandwidth than the A object. This demonstrates that Admux realizes a dynamic bandwidth allocation based on the application events and interactions.

5) Multiple-Buffer Scheme: Another simulation we conducted was to examine the effectiveness of the multiple-buffer scheme with two media streams (H object and V object). Fig. 8 shows the results of these simulations. We compare the network throughput with both the single- and multiple-buffer scheme. As we start injecting the delay, jitter, and packet loss

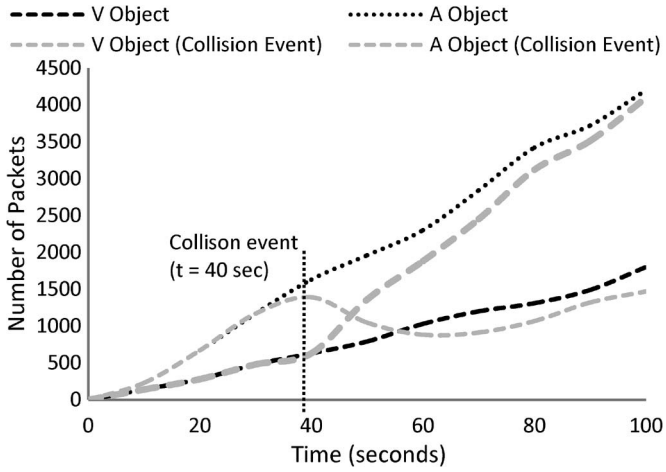

Fig. 8. Packet throughput with simulated collision event.

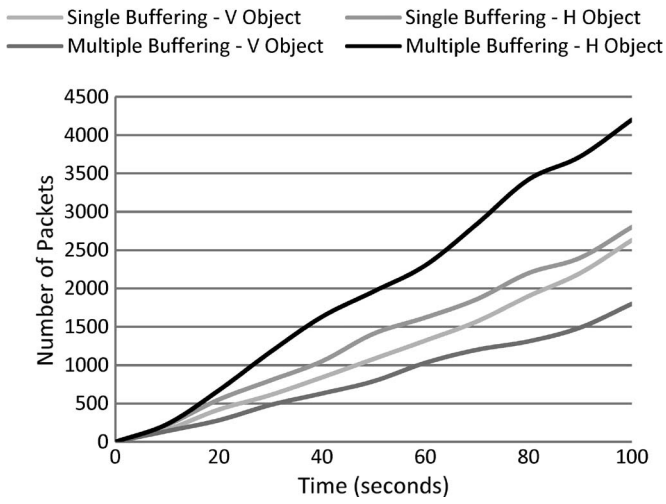

Fig. 9. Packet throughputs with single buffering and multiple buffering.

into the simulation, the resulting key update throughput for both approaches (single and multiple buffers) starts to diverge, particularly for the haptic stream (see Fig. 9).

The difference in the performance can be explained as follows: In the single-buffering approach, the retransmission buffer can easily fill up with obsolete updates under jittery and lossy conditions, whereas, in the multiple-buffering approach, unwanted key updates are strategically discarded according to the properties of the media channel to which they belong. Transferring the obsolete updates steals the available bandwidth from the fresher ones, particularly if these updates end up making it to the receiver side in the first transmission but are not acknowledged on time because of the jitter in the network. Moreover, as the retransmission buffer grows bigger, it starts to compete for the bandwidth of the sending buffer.

6) Media Channel Scalability: The scalability test is performed to examine the impact of an increase in the number of input media on the computation time of the multiplexer. We ran the same simulation while changing the number of input media channels and computed the average computation time for 1000 cycles. It is found that the average computation

time is maintained with an average of 1.83 μs. Therefore, the multiplexer performance is not affected by the increase in the number of input channels.

C. Result Discussion

The simulation results presented in this paper show that Admux outperforms its competitors in three main tracks. First, compared to existing multimedia frameworks such as MPEG-4, Admux has the capability of adapting to different network conditions while having the higher priority streams marginally affected by these changes. This is of particular importance when haptic media is used since the haptic device stability is very sensitive to changes in the network conditions.

Comparing Admux with transport-layer protocols, Admux has the distinguished feature of synchronizing haptic data with audio/video data. To the best of our knowledge, existing transport protocols cannot adapt the communication scheme based on application interactions and events. Admux has successfully been able to communicate synchronous haptic–audio–video data that are verified with the HugMe system simulation.

Finally, compared to traditional statistical multiplexing, Admux adapts to application level events and interactions. Traditional statistical multiplexing does not consider application-specific events for the multiplexing scheme. On the contrary, as shown in Fig. 7, Admux has changed the allocated resources dynamically based on the collision event. Therefore, Admux is suitable for the Internet network where QoS conditions are sporadically changing. This is achieved by strategically locating the statistical multiplexing at the application layer.

However, for dedicated networks where network resources are not significantly changing, Admux would be redundant since multiplexing can be placed and performed at the transport level.

V. Conclusion and Future Work

This paper has presented the design, implementation, and evaluation of the Admux communication framework for haptic–audio–visual data. The design, mathematical modeling, and implementation of an adaptive multiplexing scheme have been also proposed. Furthermore, an interpersonal communication system, named the HugMe system, has been simulated to test the performance of Admux. The performance evaluation has shown that Admux adapts to changes in the network conditions as well as the application interactions. Finally, the time complexity has showed that Admux operates comfortably within the 1-ms delay for the haptic interaction (1.83 μs).

As for future work, we plan to plug Admux into the HugMe system and test the networked application using the Internet network. Furthermore, we would like to test the application with users and study the immunity of Admux to changes in the network conditions and the usability implications of these changes. Finally, applications from different fields (other than chatting) will be considered for the Admux performance. Such studies will experimentally prove the ability of Admux to adapt to various haptic applications with different networking requirements and interaction styles.

References

[1] S. Oviatt, "Ten myths of multimodal interaction," *Commun. ACM*, vol. 42, no. 11, pp. 75–81, Nov. 1999.

[2] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE Trans. Inf. Syst.*, vol. E77-D, no. 12, pp. 1321–1329, Dec. 1994.

[3] A. State, D. T. Chen, C. Tector, A. Brandt, H. Chen, R. Ohbuchi, M. Bajura, and H. Fuchs, "Case study: Observing a volume-rendered fetus within a pregnant patient," in *Proc. IEEE Visualization*, 1994, pp. 364–368.

[4] "Princeton electronic billboard develops," "Virtual Arena", Technology for Television, Washington, DC: Nat. Assoc. Broadcasters1994.

[5] M. Eid, M. Mansour, A. El Saddik, and R. Iglesias, "A haptic multimedia handwriting learning system," in *Proc. Int. Workshop Educ. Multimed.*, 2007, pp. 103–108.

[6] M. Eid, M. Orozco, and A. El Saddik, "A guided tour in haptic audio visual environment and applications," *Int. J. Adv. Media Commun.*, vol. 1, no. 3, pp. 265–297, Jun. 2007.

[7] Dec. 9, 2009. [Online]. Available: http://www.trisenx.com/scent_dome.html

[8] R. MacLaverty and I. Defee, "Multimodal interaction in multimedia applications," in *Proc. IEEE 1st Workshop Multimed. Signal Process.*, 1997, pp. 25–30.

[9] D. Miras, "A survey of network QoS needs of advanced Internet applications," Internet 2 QoS Working Group2002.

[10] A. Marshall, K. M. Yap, and W. Yu, "Providing QoS for networked peers in distributed haptic virtual environments," *Adv. Multimed.*, vol. 2008, p. 841 590, 2008.

[11] T. E. Whalen, S. Noel, and J. Stewart, "Measuring the human side of virtual reality," in *Proc. Int. Symp. Virtual Environ., Human-Comput. Interfaces, Meas. Syst.*, 2003, pp. 8–12.

[12] D. Gracanin, Y. Zhou, and L. A. DaSilva, "Quality of service for networked virtual environments," *IEEE Commun. Mag.*, vol. 12, no. 1, pp. 37–51, Feb. 2004.

[13] C. Basdogan, C. H. Ho, M. A. Srinivasan, and M. Slater, "An experimental study on the role of touch in shared virtual environments," *ACM Trans. Comput.-Human Interactions*, vol. 7, no. 4, pp. 443–460, Dec. 2000.

[14] J. Kim, H. Kim, B. K. Tay, M. Muniyandi, J. Jordan, J. Mortensen, M. Oliveira, M. Slater, and M. A. Srinivasan, "Transatlantic touch: A study of haptic collaboration over long distance," *Presence—Special Issue: Collaborative Virtual Environments*, vol. 13, no. 3, pp. 328–337, Jun. 2004.

[15] R. Iglesias, E. Prada, A. Uribe, A. Garcia-Alonso, S. Casado, and T. Gutierez, "Assembly simulation on collaborative haptic virtual environments," in *Proc. 15th Int. Conf. Central Eur. Comput. Graphics, Vis. Comput. Vis.*, 2007, pp. 241–248.

[16] K. S. Park and R. V. Kenyon, "Effects of network characteristics on human performance in a collaborative virtual environment virtual reality," in *Proc. IEEE Virtual Reality*, 1999, pp. 104–111.

[17] S. Lee and J. Kim, "Transparency analysis and delay compensation scheme for haptic-based networked virtual environments," *Comput. Commun.*, vol. 32, no. 5, pp. 992–999, Mar. 2009.

[18] O. Wongwirat, S. Ohara, and N. Chotikakamthorn, "An adaptive buffer control using moving average smoothing technique for haptic media synchronization," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2005, vol. 2, pp. 1334–1340.

[19] P. Hinterseer and E. Steinbach, "A psychophysically motivated compression approach for 3D haptic data," in *Proc. Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst.*, 2006, pp. 35–41.

[20] C. Shahabi, A. Ortega, and M. R. Kolahdouzan, "A comparison of different haptic compression techniques," in *Proc. IEEE Int. Conf. Multimed. Expo*, 2002, vol. 1, pp. 657–660.

[21] P. Hinterseer, E. Steinbach, and S. Chaudhuri, "Perception-based compression of haptic data streams using Kalman filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2006, pp. 23–24.

[22] N. Sakr, J. Zhou, N. D. Georganas, J. Zhao, and X. Shen, "Prediction-based haptic data reduction and compression in tele-mentoring systems," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, 2008, pp. 1828–1832.

[23] N. Sakr, J. Zhou, N. D. Georganas, J. Zhao, and E. M. Petriu, "Robust perception-based data reduction and transmission in telehaptic systems," in *Proc. World Haptics, 3rd Joint Eurohaptics Conf. Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst.*, 2009, pp. 214–219.

[24] M. Mauve, V. Hilt, C. Kuhmunch, and W. Effelsberg, "RTP/I—Toward a common application level protocol for distributed interactive media," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 152–161, Mar. 2001.

[25] R. J. Hubbold, "Collaborative stretcher carrying: A case study," in *Proc. Workshop Virtual Environ.*, 2002, pp. 7–12.

[26] L. Gautier, C. Diot, and J. Kurose, "End-to-end transmission control mechanisms for multiparty interactive applications on the Internet," in *Proc. 8th Annu. Joint Conf. IEEE Comput. Commun. Soc. INFOCOM*, 1999, vol. 3, pp. 1470–1479.
[27] S. Dodeller, "Transport layer protocols for haptic virtual environments," M.S. thesis, Univ. Ottawa, Ottawa, ON, Canada, 2004.
[28] Z. Cen, M. W. Mutka, D. Zhu, and N. Xi, "Supermedia transport for teleoperations over overlay networks," in *Proc. IFIP-TC6 Netw. Conf.*, 2005, pp. 1409–1412.
[29] O. Wongwirat, S. Ohara, and N. Chotikakamthorn, "An adaptive buffer control using moving average smoothing technique for haptic media synchronization," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2005, vol. 2, pp. 1334–1340.
[30] S. Lee, S. Moon, and J. Kim, "A network-adaptive transport scheme for haptic-based collaborative virtual environments," in *Proc. 5th ACM SIGCOMM Workshop Netw. Syst. Support Games*, 2006, vol. 13, p. 13.
[31] Z. C. Mutka, M. Y. Liu, A. Goradia, and N. Xi, "QoS management of supermedia enhanced teleoperation via overlay networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 1630–1635.
[32] K. Chandra, *Wiley Encyclopedia of Telecommunications*. Hoboken, NJ: Wiley, 2003, ch. Statistical Multiplexing.
[33] M. A. Saleh, I. W. Habib, and T. N. Saadawi, "Simulation analysis of a communication link with statistically multiplexed bursty voice sources," *IEEE J. Sel. Areas Commun.*, vol. 11, no. 3, pp. 432–442, Apr. 1993.
[34] M. Zajeganovic-Ivancic, I. S. Reljin, and B. D. Reljin, "Video multicoder with neural network control," in *Proc. 9th Symp. Neural Netw. Appl. Elect. Eng.*, 2008, pp. 187–191.
[35] I. S. Reljin, M. Stanojevic, and B. D. Reljin, "Modified round-robin scheduler for pareto traffic streams," in *Proc. TELSIKS Conf.*, 2001, pp. 19–21.
[36] C. Liu, Y. Xie, M. J. Lee, and T. N. Saadawi, "Multipoint multimedia teleconference system with adaptive synchronization," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1422–1435, Sep. 1998.
[37] M. Eid, J. Cha, and A. El Saddik, "An adaptive multiplexer for multimodal data communication," in *Proc. HAVE Conf.*, 2009, pp. 111–116.
[38] F. R. El-Far, M. Eid, M. Orozco, and A. El Saddik, "Haptic application meta-language," in *Proc. DS-RT Conf.*, 2006, pp. 261–264.
[39] R. Gvili, A. Kaplan, E. Ofek, and G. Yahav, "Depth keying," in *Proc. SPIE Conf.*, 2003, pp. 48–55.
[40] M. Mattsson and J. Bosch, "Characterizing stability in evolving frameworks," in *Proc. 29th Int. Conf. Technol. Object-Oriented Languages Syst.*, 1999, pp. 118–130.

Mohamad Eid received the Ph.D. degree in electrical and computer engineering from the University of Ottawa, Ottawa, ON, Canada, in 2010.

He is currently a Teaching Associate with the University of Ottawa, Ottawa, ON, Canada. His current research interests include haptic multimedia applications and gaming. He has also worked with haptic modeling language and haptic application development frameworks.

Jongeun Cha received the M.S. and Ph.D. degrees in mechatronics from the Gwangju Institute of Science and Technology, Gwangju, Korea.

He is currently a Postdoctoral Fellow with the School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada. His research interests include haptic rendering algorithms, haptic interactions in broadcasting and augmented/mixed reality, haptic content authoring, and interpersonal haptic communication.

Abdulmotaleb El Saddik (M'02–SM'03–F'09) received the Ph.D. degree in electrical engineering from the Darmstadt University of Technology, Darmstadt, Germany, in 2001.

He is currently a Full Professor and the Director of the Multimedia Communications Research Laboratory with the University of Ottawa, Ottawa, ON, Canada. He is a leading researcher in haptics, service-oriented architectures, collaborative environments, and ambient interactive media and communications. He has authored and coauthored two books and more than 275 publications.

Dr. El Saddik is a Senior Member of the Association for Computing Machinery (ACM), an IEEE Distinguished Lecturer, a Fellow of the Canadian Academy of Engineering, and a Fellow of the Engineering Institute of Canada. He is the recipient of the Professional of the Year Award (2008), Friedrich Wilhelm–Bessel Research Award from Germany's Alexander von Humboldt Foundation (2007), Premier's Research Excellence Award (2004), and National Capital Institute of Telecommunications New Professorship Incentive Award (2004). He was a Theme Coleader in the Learning Object Repositories Network, Natural Sciences and Engineering Research Council of Canada (2002–2007), and a Director of the Information Technology Cluster, Ontario Research Network on Electronic Commerce (2005–2008). He is an Associate Editor of the *ACM Transactions on Multimedia Computing, Communications, and Applications*, IEEE TRANSACTIONS ON MULTIMEDIA, and IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES and a Guest Editor for several IEEE Transactions and Journals. He has been serving on several technical program committees of numerous IEEE and ACM events. He has been the General Chair and/or Technical Program Chair of more than 20 international conferences symposia and workshops on collaborative haptic–audio–visual environments, multimedia communications, and instrumentation and measurement. He was the General Cochair of ACM Multimedia 2008. His was a recipient of the Best Paper Award for three times. He has received research grants and contracts totaling more than $15 million and has supervised more than 90 researchers.