NOTE TO USERS

This reproduction is the best copy available.

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Mohamad Ahmad Eid

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Electrical and Computer Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Admux:
An Adaptive Multiplexing Framework for Haptic-audio-visual Communication

TITRE DE LA THÈSE / TITLE OF THESIS

Abudlmotaleb El Saddik

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Dorina Petriu

Shervin Shirmohammadi

Jiying Zhao

Balakrishnan Prabhakaran
(University of Texas at Dallas)

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Admux: An Adaptive Multiplexing Framework for Haptic-Audio-Visual Communication

By

Mohamad Ahmad Eid

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph. D. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-69131-1
Our file Notre référence
ISBN: 978-0-494-69131-1

Abstract

Recent trends in multimedia applications strive to incorporate multi-modal media, such as audio, video, graphics, and haptics to enhance the user's experience. Traditionally, graphic images, audio, video, text and animations define the contents used in a multimedia system. Recently, researchers have made significant progress in advanced multimedia systems by incorporating virtual reality environments, haptics, and scent into the human computer interaction paradigm. However, each media is characterized by different and sometimes conflicting communication requirements (QoS requirements) that make the communication of multiple media data a real challenge.

This thesis proposes Admux, an adaptive multiplexing framework and communication protocol for multimedia applications incorporating haptic, visual, auditory, and scent data for non-dedicated networks. First, Admux provides a standard description scheme for haptic-audio-visual applications using the Haptic Applications Meta Language (HAML) to grant the application an abstract access to the networking resources. Second, Admux uses a highly adaptive multiplexing scheme that adapts according to the application requirements, the media type (haptic, audio, video, etc), and the network conditions. Finally, the proposed framework enables dynamic media prioritization based on the application requirements and events. The simulation, as well as the implementation, evaluations have shown that Admux provides dynamic bandwidth allocation based on the network conditions, media type, and application events.

Acknowledgments

I would like to thank my supervisor Prof. Abdulmotaleb El Saddik for his enlightening advices and guidance, without which this work would not have been possible. The comments and time given by my colleague and reader Dr. Jongeun Cha have greatly improved and clarified this work. I greatly acknowledge his help for the valuable comments, suggestions and countless hours of discussions. Warm love and support from my parents is the main factor of success throughout all years of study. Lastly, praise goes to Allah whose help has given me the strength needed to complete this work.

To them all, I say thank you!

To my MOM

•

•

•

And to the M1.5 Palestinians seized in Gaza

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Multimedia Applications Incorporating Haptic, Audio, and Video Media

Multimedia utilizes multiple sensory channels to convey information that is both spatially and temporally correlated, to a user. Traditionally, graphic images, audio, video, text and animations define the contents used in a multimedia system. Recently, researchers have made significant progress in multimedia systems by incorporating virtual reality augmentations (3D virtual objects) as well as advanced media such as haptics and scent into the human computer interaction paradigm [1].

Augmenting a real scene captured from a real environment 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 and training to military. In the medical domain, for instance, the ultrasound technique facilitates the view of a volumetric rendered image of the fetus overlaid on the abdomen of the pregnant woman [3]. Another example from the entertainment domain is to place a 3D advertisement on the outfield wall of a stadium while broadcasting a baseball game [4]. Therefore, virtual objects are becoming essential part in multimedia systems and applications.

Haptics plays a prominent role in making virtual objects physically palpable in a collaborative and/or shared virtual environment. The incorporation of the sense of touch in multimedia applications gives rise to far more exciting and appealing ways of supporting collaboration, co-presence, and togetherness in these multimedia systems by

enabling users to feel each other's presence and the environments in which they are interacting [5]. The authors in [6] reviewed the applications of haptics in various multimedia applications and systems, such as medicine and training, data visualization and arts, and education and entertainment. Therefore, multimedia systems are striving towards human-computer interaction paradigm that incorporates the sense of touch.

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]. This device allows buyers to sample online perfumes, mechanics to learn to identify defects by the smell of a burnt wire, gamers to smell the sulfur before the danger is in sight, etc. Indeed, the sense of smell is envisioned to be a 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. Indeed, a multimedia application is seen as a composition of multiple streams that must be stored, disseminated, and presented in both spatial and temporal correlation. The relationships between the different media, how they interact with each other, and how a human perceives them have been studied by many researchers [1]. On the other hand, the communication of such multimedia information over non-dedicated networks such as the Internet remains a challenging research question. This thesis presents a multimedia communication framework that provides adaptive multiplexing scheme that adapts to both the application requirements as well as the network requirements.

1.2 Application scenario: Synchronous Haptic Teleconferencing System

As an application scenario, assume a child is crying (let's say in a daycare) while his parents are away. What would a child need to stop crying other than a hug and a kiss from one of his/her parent? As shown in Figure 1.1, the child is wearing a haptic suite (haptic jacket) that is capable of simulating nurture touching. The parent, on the other side of the network, uses a haptic device to communicate his feelings with his child. A 2.5D camera [5] is used to capture the image and depth information of the child and send it to the parent. The parent can touch the child captured with 2.5D camera, the touch information is calculated and sent to the child, and the child feels the touch via the haptic jacket. Whenever a collision is detected, a notification is sent from the parent host to the child host in order to activate the appropriate actuators embedded in the haptic jacket. Meanwhile, the force feedback of the child body is displayed to the parent using the haptic device.

Technically, the above scenario suggests the design and development of a haptic audio-visual teleconferencing system to enhance the physical intimacy in the remote interaction between participants. The haptic audio-visual teleconferencing system should work with a tolerable bandwidth (30-60Hz) for haptic data yet provides synchronous interaction. In this scenario, security is out of scope and not considered as a requirement even though in reality it is key for the participants' privacy.

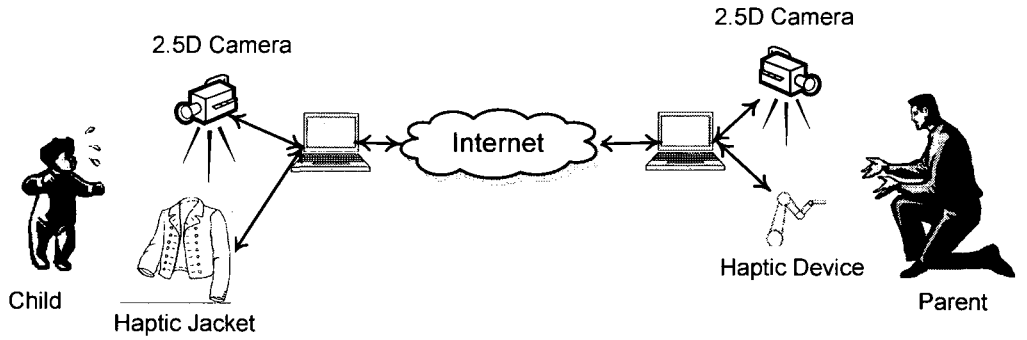

Figure 1.1 Synchronous haptic teleconferencing system.

Several requirements can be derived from the above scenario. First of all, the communication of multimedia data should be adaptable to the network conditions, the interaction events, and the media type. For instance, when the network resources are limited, the communication protocol should send only what is necessary and sufficient for the communication. Furthermore, when one of the parents is touching her/his child, the haptic media data becomes crucial and thus should be given higher communication priority than audio and video (media adaptability). Furthermore, different haptic messages should be assigned different priorities as per the communication requirements, depending on the application. For instance, in an affection session, a hug message should be given more network resources than a tap message.

Second, in order to enforce prioritization of media channel, a multiplexing scheme is needed to support higher utilization of network resources and faster communication. Finally, the application is assumed to use the Internet (a non-dedicated network) since dedicated networks are far too expensive for the users to use for social and inter-personal communication, and not as available as the Internet. Based on the

above scenario we derive five major requirements that the communication protocol should satisfy:

- *Requirement 1: Interaction-based Adaptability:* The communication protocol should adapt according to the application events and context. Events (i.e., collision) happening in the application should be used to re-allocate resources to different media data.
- *Requirement 2: Media Prioritization:* Different media data have by nature different perceptual weights as for the quality of experience of the user. Therefore, the communication protocol should support media prioritization so that each media receives resources proportional to the quality of perception. Media prioritization scheme might be static and could be read from an application description file.
- *Requirement 3: Statistical Multiplexing Scheme:* It is necessary to include statistical multiplexing for more efficient use of the network resources. Additionally, compared to mathematical multiplexing, statistical multiplexing results in lower average delays per channel.
- *Requirement 4: Internet-based:* In fact, using the Internet is a requirement that is needed for interpersonal and social interaction systems, where users do not afford the expenses of dedicated networks. This assumption is valid since Internet is highly available compared to dedicated networks and have much lower connection cost.
- *Requirement 5: Network Adaptability:* As a consequence of the previous requirement (Internet-based), the network resources become more limited and

changing. Therefore, the communication protocol should adapt to the network conditions and changes.

1.3 Research Statement

The communication of multimedia data poses several challenges. First, being Internet-based, the network resources are limited by nature and dynamically changing. Second, in a multimedia application, not every media has the same contribution weight for the user perception. For example, the degradation in the haptic rendering (in case of extensive delays and/or jitters) may result in severe loss of quality and instability of the haptic device. In fact, several haptics data communication researches have shown how even small delays can affect task completion time and how jitter has a greater impact on predicting other's actions [8]. Finally in a multimedia application, data of specific events (such as collision data) must be given higher precedence than other data in accordance with the application specifications and requirements.

On the other hand, each media channel has different stream characteristics (such as data unit size and data rate) that pose another communication challenge. For instance, haptic data is very short in size (usually position or force information with 6 bytes per frame) but should be transmitted at very high rates (around 1 kHz). The communication of haptic samples on a 1 packet per sample basis results in an inappropriate payload/header ratio, which wastes expensive network resources due to increased protocol overhead. On the other hand, video data is communicated at only 60 Hz rate whereas the size of the transmitted frame is very large (around 200 bytes). To the best knowledge of the authors, there is no application-layer multiplexing scheme that can optimize the header/payload ratio so that higher bandwidth utilization can be achieved.

Finally, there might be very different and sometimes conflicting QoS parameters associated with the communication of each media channel. Therefore, a communication protocol is desirable to enforce QoS requirements based on the corresponding media type.

In this thesis, we identify the research statement as the following:

"*Developing an adaptive application layer communication protocol for multimedia applications incorporating haptic, visual, auditory, and scent information, among other media, for non-dedicated networks*".

1.4 Thesis Contributions and Goals

The principal objective of this work is to design, implement, and evaluate an application layer communication framework for multimedia data, incorporating haptic media, over non-dedicated networks (specifically the Internet). The thesis contributions can be summarized as follows:

- design and development of a haptic application meta-language – named HAML – as a standard by which haptic application components such as haptic devices, haptic APIs, or graphic models make their specifications and capabilities self-described.
- design and development of a multimedia communication framework that adapts to both network conditions and application requirements.
- design and analysis of a mathematical model for an adaptive multiplexing scheme (named Admux) to optimize the communication of multimedia contents including haptic, audio, video, and graphic data.

- design and development of a communication protocol that realizes the framework operation. The protocol defines the syntax and semantics of the communicated data packets with header information to enforce multiplexing and prioritization rules.

- design and implementation of a proof-of-concept application (named HugMe system) to show the potential of the developed framework to optimize the communication of multimedia data.

Therefore, the proposed work comprises the following distinguished features:

- *Adaptability:* The proposed protocol adapts to the application requirements, the media type, and the network resources therefore it is implemented at the application layer. To further improve the customizability of the protocol the quality of service requirements, associated with a specific multi-modal application, are defined in HAML format (such as the average delays, jitters, packet loss, and packet rate, per each input channel). The protocol parses the HAML file and sets up the communication session accordingly.

- Media Prioritization: The proposed protocol adapts to the application dynamics (such as operation modes and application specific events) and to changes in the network conditions and resources. The protocol measures periodically the available network resources and adapts the resources allocated to each input media channel accordingly.

- Multiplexing scheme: The proposed protocol intelligently multiplex the input media channels into a continuous stream of data using media prioritization. The available resources are distributed to the input channels based on the respective quality of service

requirements associated with each media channel. It is worth mentioning that the media priorities are application and communication dependent so they can dynamically change during the application operation.

1.5 Thesis Organization

The thesis is organized as follows. In chapter 2, we review communication protocols for multimedia applications with multiple media and compare each against the requirements set we developed in Chapter 1. Chapter 3 introduces the Haptic Application Meta Language (HAML) and how it is utilized as a description scheme for the application to communicate its requirements with the communication framework. The proposed communication framework and protocol is the subject of chapter 4. We present the protocol design philosophy, the distinguished features, the communication framework, and the communication protocol. Chapter 5 presents the core of this thesis; the adaptive multiplexer with its mathematical model and simulation analysis and results. Chapter 6 presents the development of a proof-of-concept application (the application scenario presented in this chapter) and the performance evaluation of the proposed protocol under various network conditions. We summarize the thesis contents and contributions in Chapter 6 and provide suggestions for future research directions.

1.6 Scholastic Output and Achievements

Journal papers:

1. **Mohamad Eid**, Jongeun Cha, and Abdulmotaleb El Saddik, "Admux: An Adaptive Multiplexer for Haptic Audio Visual Data Communication", IEEE Transactions on Instrumentation and Measurement, (accepted to appear).

2. Jongeun Cha, **Mohamad Eid**, and Abdulmotaleb El Saddik, "Touchable 3D Video System," ACM Transactions on Multimedia Computing, Communications and Applications (ACM TOMCCAP), Vol. 5 (4) pp: 1-25, 2009
3. Ahmad Barghout, Atif Alamri, **Mohamad Eid**, Abdulmotaleb El Saddik, "Haptic Rehabilitation Exercises Performance Evaluation Using Automated Inference Systems", Int. Journal of Advanced Media and Communication, Vol. 3(2), pp: 197 – 214, 2009.
4. Atif Alamri, **Mohammad Eid**, Rosa Iglesias, Abdulmotaleb El Saddik, and Shervin Shirmohammadi, "Haptic Virtual Rehabilitation Exercises for post-stroke diagnosis", IEEE Transactions on Instrumentation and Measurements Vol. 57 (9), pp.: 1876-1884, 2008
5. **Mohamad Eid**, Mauricio Orozco and Abdulmotaleb El Saddik, "A Guided Tour in Haptic Audio Visual Environment and Applications ", Int. J. of Advanced Media and Communication, vol.10, no.1, pp.10-17, Feb. 2007.

Conference papers:

1. **Mohamad Eid**, Jonguen Cha, and Abdelmotaleb El Saddik, "An Adaptive Multiplexer for Multi-Modal Data Communication", IEEE Workshop on Haptic Audio Visual Environments and Games, Lecco, Italy, 2009.
2. Jongeun Cha, **Mohamad Eid**, Ahmad Barghout, ASM Mahfujur Rahman, and Abdulmotaleb El Saddik, "HugMe: Synchronous Haptic Teleconferencing," *ACM International Conference on Multimedia, The Multimedia Grand Challenge*, pp. 1135-1136, Beijing, China, Oct. 19-24, 2009, (**ACM MM most Entertaining Award**).
3. **Mohamad Eid**, Jongeun Cha, and Abdulmotaleb El Saddik, "HugMe: A Haptic Videoconferencing System for Interpersonal Communication," *IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems (VECIMS)*, pp. 5-9, Istanbul, Turkey, July 14-16, 2008.
4. Abu Saleh Md. Mahfujur Rahman, **Mohamad Eid**, and Abdulmotaleb El Saddik, "KissMe: Bringing Virtual Events to the Real World", In proceedings of the 2008 IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Turkey, 2008.
5. Hussein Al Osman, **Mohamad Eid**, and Abdulmotaleb El Saddik. Evaluating ALPHAN: A Network Protocol for Haptic Interaction. 2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Page(s):361 – 366, 2008.
6. Jongeun Cha, **Mohamad Eid**, and Abdulmotaleb El Saddik, "DIBHR: Depth Image-Based Haptic Rendering", In Proc. of the EuroHaptics 2008 conference, pp. 640 – 650, June 12-14, Madrid, Spain
7. Hussein Al Osman, **Mohamad Eid**, and Abdulmotaleb El Saddik, "Evaluating ALPHAN with Multi-user Collaboration", 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, Page(s):181 – 186, 2008, (**Best Paper Award**).
8. Jongeun Cha, **Mohamad Eid**, Lara Rahal, and Abdulmotaleb El Saddik, "HugMe: An Interpersonal Haptic Communication System," *IEEE International*

Workshop on Haptic Audio Visual Environments and their Applications (HAVE2008), pp. 99-102, Ottawa, Canada, Oct. 18-19, 2008.

9. **Mohamad Eid**, Sheldon Andrews, Atif Alamri, and Abdulmotaleb El Saddik, "HAMLAT: A HAML-based Authoring Tool for Haptic Application Development", In Proc. of the EuroHaptics 2008 conference, pp. 857-866, June 12-14, Madrid, Spain
10. Abdelwahab Hamam, **Mohamad Eid**, Abdulmotaleb El Saddik, and Nicolas Georganas "A Fuzzy Logic System for Evaluating Quality of Experience of Haptic-Based Applications", In Proc. of the EuroHaptics 2008 conference, pp. 129-138, June 12-14, Madrid, Spain
11. Abdelwahab Hamam , **Mohamad Eid**, Abdulmotaleb El Saddik and Nicolas D. Georganas, "A Quality of Experience Model for Haptic User Interfaces" In Proc. of Haptic in Ambient Systems (HAS 2008) Workshop, Feb. 11-14, 2008, Quebec City, Canada.
12. **Mohamad Eid**, Atif Alamri, Jamil Melhem and Abdulmotaleb El Saddik, "Evalutation of UML CASE Tool with Haptics" In Proc. of Haptic in Ambient Systems (HAS 2008) Workshop, Feb. 11-14, 2008, Quebec City, Canada.
13. **Mohammad Eid**, Mohamed Mansour, Rosa Iglesias, and Abdulmotaleb El Saddik , "Haptic Multimedia Handwriting Learning System", In Proceedings of ACM Workshop on Educational Multimedia and Multimedia Education (EMME 2007), September 28, 2007, Augsburg, Germany
14. Hussein Al Osman, **Mohamad Eid**, Rosa Iglesias, and Abdulmotaleb El Saddik, ALPHAN: Application Layer Protocol for HAptic Networking, Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, October 2007.
15. Sheldon Andrews, **Mohamad Eid**, Atif Alamri and Abdulmotaleb El Saddik, "Extending Blender: Development of a Haptic Authoring Tool" Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, October 2007.
16. A. Alamri, **M. Eid**, A. El Saddik, "A Haptic-enabled UML CASE Tool", in Proc. 2007 International Conference on Multimedia & Expo (ICME), July 2-5, 2007, Beijing, China.
17. **Mohamad Eid**, Mohamed Mansour, Rosa Iglesias, and Abdulmotaleb El Saddik, "A Device Independent Haptic Player" In proceedings of the 2007 IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Italy, June 2007.

Patents:

1. Abdulmotaleb El Saddik, Jongeun Cha, **Mohamad Eid**, Fawaz A Alsulaiman, Atif Alamri, Lara Rahal, Daniel J. Martin, "A Synchronous Interpersonal Haptic Communication System," **Patent Pending**, Application number: 12/481,274, Date: 2009.06.09

CHAPTER 2

BACKGROUND AND RELATED WORK

When two humans communicate in a face-to-face fashion, they do so by exchanging information via many several channels or modalities. 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 socio-emotional aspects of the communication [9]. 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 multi-modal communication and this is why it is the most perceivable [9].

2.1 Introduction

2.1.1 Multimedia Communication System

A multimedia communication system transports multiple media information such as audio, video, and haptics over a computer network. Figure 2.1 shows the one-way structure diagram of a typical multimedia communication system with N media input channels. The input channels are seen as continuous streams of media data that undergo processing pipelines before transmission. First, they are converted into a digital form (analog-to-digital conversion), then encoded and/or compressed, then multiplexed and sent over one transport channel. At the receiver side, the multimedia data is de-

multiplexed to the corresponding media channels, de-compressed, and rendered to the user through a particular display interface.

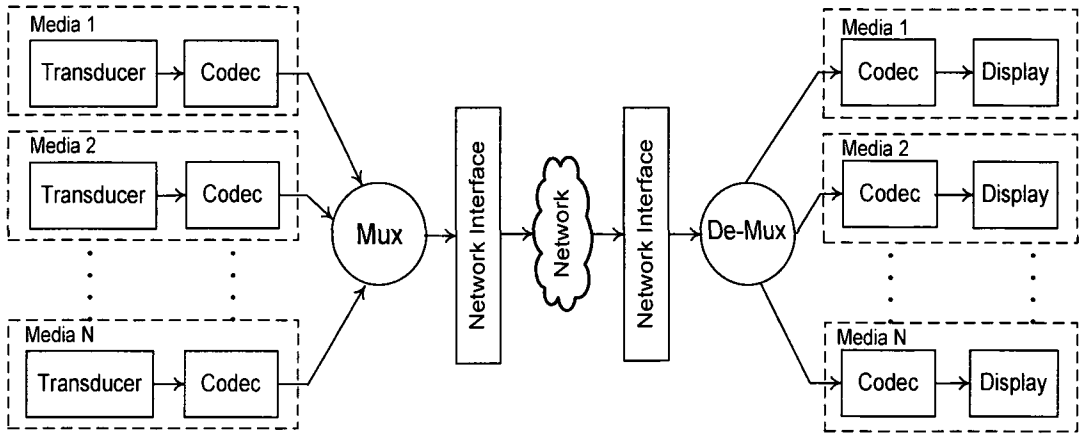

Figure 2.1: A general multimedia communication framework.

The components of the multimedia communication system are described briefly in the followings:

- *Media Transducer*: this component converts the captured media information into a form readable by the computers. Examples of media transducers include a microphone, a camera, haptic device, among others.

- *Media Codec*: this component processes the raw data received from the media transducer. The processing of this information includes data filtration, data encoding, and data compression. There are many compression techniques that are designed for different media information such as audio and video compression, haptic data compression, and virtual objects compression.

- *Multiplexer*: this component provides a means to interleave data from different input media channels into one serialized bit stream. The multiplexer should

function so that the QoS requirements associated with each media channel are met or at least optimized.

- *Network Interface*: this component encapsulates the multiplexed data into communication units (packets) by adding network-related header/tail for networking needs. It also provides Admux with abstract access to the underlying transport medium.

- *De-multiplexer*: this component does exactly the opposite of what a multiplexer does. It receives a continuous stream of multimedia information and forwards the data to the appropriate media channel.

- *Media Display*: this component converts the digital media signal into a form perceivable by the end user. Examples of media displays include a computer screen, a speaker, and a haptic device, etc. Notice that the media displays must be presented to the user in a synchronous fashion, i.e. with proper spatial and temporal relationships.

2.1.2 Review of Existing Multimedia Communication Frameworks

The idea of building a framework that facilitates the development and distribution of multimedia applications found significant interest from both the research and industry communities. However, to the best of our knowledge, there is no framework for multimedia communication capable of communicating general haptic, audio, video, and graphics data that are synchronized together. Some frameworks (such as MPEG-4, MPEG-2, augmented reality frameworks, etc.) support the communication of audio, video, and graphics data but not haptic media whereas others (such as CHAI 3d, Reachin API, and H3D) focus on haptic data communication only.

MPEG-4 is an object-based multimedia framework that supports streaming data for various media as well as interactivity in broadcast multimedia applications [10]. A

part of the standard is dedicated to delivery issues by defining Delivery Multimedia Integration Framework (DMIF); a generic platform for the delivery of multimedia information. DMIF addresses three different content access scenarios: local files, broadcast, and remote interaction. In MPEG-4, the scene is compressed and streamed using MPEG-4 Binary Format for Scene (BIFS). The user can interact with objects of the scene such as navigating into 3D worlds, modifying the position of the objects, and interacting with other users. However, MPEG-4 does not support the communication of force feedback haptic data, particularly when the update rate is extremely high (around 1 kHz).

Some researchers have tried to augment MPEG-4 specifications to transmit haptic data (for example [11], [12], and [13]). For instance, the authors in [12] propose an authoring and editing framework that extends MPEG-4 BIFS to conveniently represent haptic data in the scene graph and facilitate haptic data broadcasting. New nodes for tactile and kinesthetic data are added to the BIFS description language in order to attach various haptic properties to the scene. The research presented in [13] shows that with MPEG-4 BIFS communication, the time delay effect on haptic communication is reduced.

CHAI 3D is an object oriented application framework for facilitating a developer-friendly creation of multimedia virtual worlds including haptics, visualization and interactive real-time simulation [14]. CHAI 3D integrates the haptic and graphic representations of objects and abstracts away the complexities of individual haptic devices. However, CHAI 3D does not support any communication mechanism for audio/video transmission.

Reachin API allows the development of hapto-visual applications by using programming languages such as C++, Phyton, or VRML [15]. The Reachin API embeds high-fidelity features as well as a complete set of classes, nodes and interfaces for managing and synchronizing the haptics and graphics, and audio aspects of advanced 2D and 3D applications in a hierarchical data structure. Even though Reachin API supports data communication over Internet based on the download-and-playback concept, there is no mechanism defined for live streaming and real-time interactions.

The H3D API, based on X3D, is an open-source haptics software development platform that is based on the scene-graph approach to build haptic virtual environments [16]. It is dependent on OpenGL for graphics rendering and OpenHaptics for haptic rendering. H3D provides a scene graph that performs both graphic and haptic rendering from a single scene description. Similar to Reachin API, H3D does not support any real-time communication mechanism for synchronized haptic-audio-video data.

2.1.3 Multimedia Communication Requirements

Quality of Service (QoS) quantizes the quality of information delivery as a set of quality parameters that can be, among others defined 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 [17]. In this thesis, four parameters of the QoS metrics are considered: delay, jitter (the variation of the network delay), data loss rate, and data rate. The requirements for these four QoS parameters as for the four media types addressed in this thesis (video, audio, graphics, and haptics) are discussed in this section.

Video conferencing applications are delay-sensitive and loss-sensitive by nature [17]. In order to provide a high quality video communication, the underlying transport network should guarantee the network-level QoS requirements. The QoS requirements for video communication are specified in [18] as: (1) transfer delays should always be less than or equal 400 ms to ensure an acceptable quality of perception level, (2) the jitter effect should not exceed 30 milliseconds, (3) the bandwidth requirements, compared to other media, are the most demanding (from 2.5 Mbps to 5 Mbps), and finally, (4) the data loss rate should always be within 1% for stable video rendering.

In this thesis, the term "audio" refers to any kind of sound signal, including both voice and music signals which results in higher bandwidth requirements than voice data. Depending on whether the application is interactive or not, the delay constraints can be either tight or relaxed. However, as a rule of thumb, humans expect higher fidelity audio and do not usually tolerate quality degradations [17]. Bandwidth is not a major concern for audio traffic; typical values range from 22 Kbps to 200 Kbps. The requirements for audio media are concerned mainly about the end-to-end delay, jitter and loss that should be kept within guaranteed boundaries. Typical delays in audio data transfer are bounded to 150 ms whereas jitters should be maintained at a maximum 30 milliseconds. As in video, audio data loss rate should be less than 1%.

Networked virtual environments enable users to share virtual objects over a network [19]. The quality of a virtual environment is commonly described and evaluated at the application and user levels and is usually referred to as Quality of Experience (QoE) [20]. QoE parameters include user preferences, user performance, satisfaction, etc. However, network QoS requirements have direct impact on the overall quality of experience [21]. Several studies were conducted to derive the network QoS requirements

for graphics and virtual objects communication, and the findings were as follows: transfer delay ranges from 100 ms to 300 milliseconds, jitter should be less than or equal 30 milliseconds, packet loss rate is about 10% or less, and requires a bandwidth ranging from 45 Kbps to a maximum1.2 Mbps [21].

The concept of sending haptic data over a network is referred to as "tele-haptics" [22]. One unique requirement in tele-haptics is that un-wanted 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 per the QoS requirements [25]. The latency requirement is very strict for haptic data and is largely affected by the performed task (tactile or kinesthetic feedback) and usually range between 3 milliseconds and 100 milliseconds [26-27]. It has also been shown that jitter has the greatest impact on coordination performance when the latency was high and the task was difficult (1 milliseconds to 10 milliseconds) [25]. The bandwidth requirement is more relaxed than delay and jitter requirements, it is about 128 Kbps [18]. Finally, the data loss rate can also affect the quality of haptic rendering and stability; it ranges between 0.01% and 10% [18].

2.1.4 Requirements Summary

Table 2.1 summarizes the QoS parameters associated with the four media: audio, video, graphics, and haptics. Notice that these requirements are very varying per each media as well as among the multiple media. It is also clear that haptic media is more sensitive to delay and jitter than other media types. Therefore, designing a

communication protocol that adapts to the different and varying multimedia requirements is the challenge we are investigating in this thesis.

Table 2.1 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 – 5 Mbps]	[45 Kbps – 1.2 Mbps]	[128 Kbps]

2.2 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 directions to handle this research subject: (1) improve the control mechanisms at either ends of the communication to accommodate the unpredictable behavior of the Internet such as the use of delay compensation techniques[28] and jitter smoothing algorithms[29], (2) improve the quality of service of the Internet and make it as reliable as a dedicated network is, and (3) optimize the volume of communicated data, and transmit only what is necessary and sufficient to maintain the overall quality of perception for the end users. This section presents researchers efforts in these three areas and evaluates them against the five requirements defined in Chapter 1.

2.2.1 Compression and Control

Fuelled by the demands for real-time simultaneous recording and transmission of voluminous data that are produced by multiple sensors, multimedia compression has

received significant attention. The compression research for audio, video, and graphic data has reached a matured stage (including MPEG, H.261, H.262, etc). However, despite the stringent need for haptic data compression, the field is still in its infancy and many open areas have emerged. There have been three directions with haptic data compression: (1) developing systems for real-time compression of heterogeneous haptic information [30], (2) exploring the suitability of existing compression techniques for haptic data [31], and (3) minimizing the amount of data using perception-based data reduction [26][27][32]. Furthermore, extracting semantic information from the sampled haptic data would help reducing the amount of data required to describe a session, and thus enabling efficient storage and transmission of haptic data [31].

The communication of haptic signals is characterized by strict delay constraints in order to assure local control loop stability. Therefore, haptic samples have to be transmitted immediately – to minimize the overall delay – which disfavors block-based processing that is commonly used in audio or video transmission. In case of a packet-switched network such as the Internet, this strategy quickly results in high packet rates of up to the typical sampling rate of 1 kHz. In order to enable haptic applications in the presence of significant transmission delays several control architectures have been developed to address the stability issues [33-36]. In particular, control architectures based on the scattering theory [33] are often deployed to stabilize kinesthetic systems in the presence of constant time delay. These approaches ensure the desired stability by preserving passivity within all elements of the communication system. One approach is based on Weber's Law, where insignificant changes (called the Just-Noticeable Difference (JND)) in the processed haptic data streams are unperceivable and corresponding haptic samples can be dropped. This approach is named deadband and is

presented thoroughly in [37] and [38]. The authors in [39] have extended this approach by combing the concept of event-based haptics [40] with deadband to improve the contact realism in collaborative haptic applications. In this approach, a local contact model is utilized during the events of remote/virtual contacts since the haptic signal requires increased bandwidth. This concept is similar to the work presented in this thesis as we change the allocated network resources based on the application level events and interactions.

On the other hand, control techniques are used to ensure consistency for haptic applications such as Dead Reckoning (DR). In Dead Reckoning, it is assumed that the behavior of each object over time is predefined. Thus, each object's state is updated locally. Operations are also performed locally on each object and after each local update, the resulting state is calculated with respect to the one that could have been obtained from the DR algorithm; if the difference between these two states 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 has proposed an alternative approach to solve the problem of inconsistency using the local lag and the timewrap algorithms [41]. In all, Dead-Rockening meets only two out of five requirements: the interaction-based adaptability and Internet-based.

Another common control technique is called local lag [41], where it is assumed that each operation is qualified with two timestamps: 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 timestamps is called the lag value. Choosing an appropriate lag value is important in order to balance the responsiveness of the system and its

consistency. Mauve proposed the use of a constant local lag throughout the execution of the application [41]. The local value is extracted from two values: the maximum of the average network delays among all participants and the maximum allowable response time of the application. The latter value is obtained from psychological tests and is subjective [42]. Chen proposed an adaptive approach for choosing the local lag value [43]. That is, using statistical sampling, the network latency of the next short duration is estimated and the local lag value is corrected accordingly. This approach meets the last three requirements defined in Chapter 1, namely the interaction adaptability, Internet-based, and network adaptability. However, it does not support any multiplexing scheme or media prioritization.

The variation of the network delay is known as jitter. One way to smooth the jitter has been proposed by [44] where messages are multicasted 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 timestamps and not their time of arrival. This approach has two main advantages: ease of implementation and application independence. However, these advantages come at the cost of an increased overall delay. Therefore, jitter smoothing meets only two requirements: Internet-based and network adaptability, it does neither support multimedia multiplexing/prioritization nor interaction-based adaptability.

2.2.2 Transport and Network Protocols

Few communication protocols have been designed for multimedia applications since it is difficult to capture the widely varying requirements of different media into one generic protocol. The generic transport-layer protocols (namely TCP and UDP) are used

by several multimedia applications. Few protocols have been proposed and evaluated for haptic applications (such as SCTP, Smoothed SCTP, Light TCP, RTP/I, and STRON).

TCP provides several services including error control, sequence control, loss control and duplication control 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, while network resources are being exhausted in order to re-transmit an obsolete one [45]. UDP does not suffer from any of the drawbacks of TCP however it does not fully suit the reliability requirements of multi-modal multimedia applications [45]. For instance, UDP has no buffering delays and lower jitter because it does not resend lost packets [45]. Nonetheless, UDP remains the most popular transport layer protocol for real time applications and therefore has been used as the transport protocol for many haptic data communication. UDP supports none but the Internet-based requirement.

The Synchronous Collaboration Transport Protocol (SCTP) is similar to the UDP protocol since most of the messages are sent unreliably whereas key messages are sent reliably. It also differs from UDP since the sequence numbers are used for packet ordering. For each key message, a timer is set and if a timeout occurs before receiving the acknowledgement of the message in question, it is resent, always as a key update [45]. It has been proven that for collaborative applications, SCTP, performs better than protocols with negative acknowledgments [45]. Consequently, SCTP satisfies two requirements, namely interaction-based adaptability and Internet-based.

The smoothed SCTP protocol adds a jitter smoothing mechanism to the SCTP protocol. On the sender side, the Smoothed SCTP and SCTP protocols are the same. On the receiver side, each received update is placed in a bucket according to its timestamp.

The receiver constantly checks for updates that have been sent every a given threshold in milliseconds and in case an update is found, it is retrieved from the bucket and forwarded to the application [46]. This means that all updates, including the ones generated locally, are processed with constant delays (δt). Same requirements as SCTP are met with smoothed SCTP.

Light TCP is inspired from TCP and supports the concept of message obsolescence. The sender queue accepts update messages from the application and processes them as follows [47]: (1) a key update is placed at the end of the queue and marked as a key message in order to prevent it from being erased, (2) a normal update can replace older normal updates for the same shared object, and (3) unacknowledged update messages are placed again in the queue if no newer updates from the same object have been produced by the application. At the receiver, a received update is immediately forwarded to the application if its sequence number is bigger than the last received update's sequence number, otherwise it is dropped. There is no buffering. Light TCP is interaction-based adaptable, supports message prioritization, and is Internet-based (meets three requirements out of five).

The Real-Time Protocol for Interactive applications (RTP/I) is an application layer protocol. It is designed for general network distributed interactive applications. Nonetheless, haptic applications fall under this category. The communicated messages are categorized as: event, state and request-for-event packets [48]. An event packet carries an event or a fraction of an event. A state packet carries the entire state of a subcomponent in the environment. The request-for event packets are used by participants to indicate that the transmission of a subcomponent's state is required by the sender [48].

Therefore, RTP/I is interaction-based adaptable and Internet-based (meets two of the five requirements).

The research presented in [49] proposes a framework of 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 network performance degradation. This work meets one requirement of the five defined in Chapter 1, namely the network adaptability.

The authors in [50] proposed a haptic data transport scheme that reduces the transmission rate by using adaptive aggregated packetization and a priority-based filtering. The proposed scheme adapts the transmission, loss rate, and buffering time of haptic events to the current network state based on the delay and loss effects of each haptic event. In order to offset jitter with a small playout delay, an intramedia synchronization scheme with dead-reckoning is utilized. The priority-based haptic event filtering and network-adaptive haptic event aggregation of the proposed transport scheme has resulted in a lower transmission rate than the other transport schemes (Reed-Solomon FEC error control and selective ARQ error control and congestion control).

Another protocol, named ALPHAN [51], uses a similar approach using a multiple buffer scheme to prioritize and optimize media data transfer. Additionally, ALPHAN uses HAML description language [66] to define the application requirements and pass them on to the network protocol. However, the protocol as described in [52] is not adaptable to the network resources and conditions. The protocol supports the notion of key updates that is widely supported by most of the haptic communication transport

layer protocols, by implementing an application layer reliability mechanism that is only applied to such updates, while "normal updates" remain unaffected. ALPHAN also makes use of the Multiple Buffering (MB) scheme. In this scheme, every object in the application is attributed a sending buffer. Allocating a buffer for each object permits the decoupling of update transmission for different objects, especially if they are independent from each other or they need to be prioritized based on user and/or application preferences.

2.2.3 Statistical Multiplexing Schemes

Statistical multiplexing is a proven technique used to improve the efficiency of communication over a limited bandwidth network [53]. The principle is that a group of media channels share a limited 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 channels with less QoS requirements. Most statistical multiplexing research is done at the transport layer (especially ATM networks [53]). However, at the application layer, there are very few efforts to use adaptive multiplexers for communicating multi-modal data, including haptic media [31].

One of the few works in this area is to dynamically control the arrival rate of multimedia data by switching the coders to different compression ratio (changing the coding rate) based on the network conditions [54]. 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. This work

meets three requirements out of five: it supports media prioritization, multiplexing, and it is Internet-based.

The work in [55] investigated the use of self-organizing neural networks to design a statistical multiplexer for video streams. The proposed approach sues multiple video coders, followed by a multiplexer that generates the aggregate sequence for few video streams. Three control methods are proposed: priority control, rate control, and the combination of both. The neural network approach performed very well to improve the packet loss as compared to Round Robin (RR) approach [56], and to smooth the variations of delays with rate control. Consequently, it meets two requirements, namely media prioritization and statistical multiplexing.

2.2.4 Multimedia Rate Shaping and Synchronization

Rate shaping techniques attempt to adjust the rate of audio-visual 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 frame rate at which the media is transmitted, the quantizer that defines the sampling resolution for the multimedia streams, and movement detection threshold (for inter-frame coding). The movement detection threshold limits the transmission of similar blocks from previous frames.

Adaptive synchronization can be used for multimedia teleconferencing systems involving multiple channels of communication [57]. For instance, the adaptive synchronization technique proposed in [57] is immune to clock offset and/or clock frequency drift, it does not need a global clock and provides the optimal delay and

buffering for the given QoS requirements. The adaptive synchronization technique can be used to solve both intra-media (in a single stream) synchronization and inter-media (among multiple streams) synchronization. The main idea used in the synchronization algorithm is to divide the packets into *wait*, *no wait,* and *discard* categories. Packets in the *wait* bucket are displayed after some time, *no wait* packets are displayed immediately, and *discard* category packets are discarded.

The basic adaptive synchronization algorithm requires the user to specify the acceptable synchronization error, maximum jitter and maximum loss ratio. The sender is assumed to put a timestamp 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. Achieving intra-media synchronization is a straight-forward application of the basic algorithm. For inter-media synchronization, a group PBC is used, which is incremented and decremented based on the slowest of the streams to be synchronized.

2.3 Summary of Related Work

As a summary, the surveyed related works covered partially the requirements defined in Chapter 1. None of them has met all the requirements. Therefore, our intention in this thesis is to design, develop, and evaluate a multi-modal communication protocol that satisfies all the five requirements. Table 2.2 lists a summary of related works along with which requirements are met by each.

On the other hand, the proposed protocol satisfies all the requirements listed in Table 2.2. First, it is interaction-based adaptable since the multiple buffering scheme and prioritization are based on application events. Second, the use of multiple channels enables the protocol to enforce media prioritization since each channel can be treated differently by the multiplexer. Furthermore, Admux provides adaptive statistical multiplexing of the multiple channels based on the application and network conditions which meet the statistical multiplexing scheme and network adaptability requirements. Finally, Admux is based on UDP, which means it is Internet-based.

Table 2.2: Summary of related work against five requirements.

Type	Protocol	Interaction-based Adaptability	Media Prioritization	Statistical Multiplexing Scheme	Internet-based	Network Adaptability
Compression and Control	Dead-Rockening	✓	X	X	✓	X
	Local lag	✓	X	X	✓	✓
	Jitter smoothing	X	X	X	✓	✓
Network protocols	UDP	X	X	X	✓	X
	SCTP	✓	X	X	✓	X
	Light TCP	✓	✓	X	✓	X
	RTP/I	✓	X	X	✓	X
	STRON	X	✓	X	X	✓
Multiplexing Scheme	Codecs Control	X	✓	✓	✓	✓
	Round-Robin Scheduler	X	✓	✓	X	X
	Neural Network Multiplexing	X	✓	✓	X	X
	Admux	✓	✓	✓	✓	✓

CHAPTER 3

HAPTIC APPLICATION META LANGUAGE (HAML)

The continuous evolution of computer haptics, as well as the emergence of a wide range of haptic interfaces has recently boosted the haptics domain. Even though efficient tools that support the developer's work exist, little attention is paid to the reuse and compatibility of haptic application constituents. In response to these issues, we propose an XML-based description language, namely Haptic Application Meta Language - HAML. The envisioned goal is to allow for the creation of plug-and-play environments in which a wide array of supported haptic devices can be used in a multitude of virtual environments and communication conditions, with the compatibility and adaptability issues being handled by automated engines instead of programmatically by the user.

3.1 Why HAML?

Since the release of the Novint Falcon, from Novint Technologies Inc. in 2008 that costs less than $200, haptic devices are becoming accessible for personal use [58]. Therefore, we anticipate that haptic devices will undergo wide deployment and use for personal customers in the near future, alongside with gaming industry. Currently each haptic device uses a different API and thus making application development API and device specific. Nowadays we are able to download audio or video clips, and play them back with some standard commercial players, such as RealOne Player or Windows Media Player. Since these videos are coded following certain standard formats, they are highly independent from the graphics cards or displays we are using. Similarly, HAML is meant to define a standard technology-neutral format by which haptic application components

such as haptic devices, haptic APIs, or graphic models make themselves and their capabilities known.

There have been several modeling languages, such as SensorML [59] and Transducer Markup Language (TML) [60] that can partially describe a haptic application. For instance, SensorML models a sensor or actuator as a process that has input(s) and produces output(s) based on predefined methods. SensorML could not be efficiently used to describe haptic applications for at least two reasons: first the haptic interface is characterized by bi-directional flow of data/energy where the division between "input" and "output" is often very fine and difficult to define, and second SensorML does not provide description for the mechanical design and behavior of the device – such as applied forces and workspace dimensions.

Furthermore, virtual environment modeling languages such as VRML [61] and Web3D Consortium's X3D [62] fall short too in describing the haptic interface hardware and consequently, tailoring the virtual environment to fit a particular haptic device is tiresome, low-level, and programmatic endeavor. Furthermore, neither VRML nor X3D provide descriptions for communication specifications such as QoS network requirements. The goal of HAML is to intuitively solve this problem by providing a comprehensive description that enables an interpretive backend engine to discern and solve compatibility issues and adapts to available devices and resources.

Many researchers realized the need for a formal standard description language for haptic models. For instance, the work in [63] proposed a novel XML-based approach to represent generic haptic application. The model included: application general information, haptic interface, haptic and visual rendering, and the system behavior, among others. Many key features related to tactile and haptic interaction were not

covered. Meanwhile, an ongoing effort to introduce a work plan for the development of a new set of ISO standards for tactile and haptic interactions, based on the GOTHI model [64], has been described in [65]. These standards will provide ergonomic requirements and recommendations for haptic and tactile hardware and software, and guidance related to the design and evaluation of hardware, software, and combinations of hardware and software interactions. Even though the proposed draft for the standard provides comprehensive guidelines for haptic interactions, it lacks the description of the application-specific features, the virtual environment, and the quality of experience specifications (including QoS requirements).

Conversely, HAML is designed to provide a technology-neutral description of haptic models. It contains ergonomic requirements and specifications for haptic hardware and software interactions. In other words, HAML is the standard by which haptic application components such as haptic devices, haptic APIs, or graphic models make themselves and their capabilities known. Thus the purpose of HAML is to:

1. Provide a standard technology-neutral description language for haptic application components
2. Provide general haptic information in support of device/component discovery
3. Support a standard for haptic data representation
4. Support the processing and analysis of haptic data
5. Solve the incompatibility issues between different haptic devices and APIs
6. Capture application-specific requirements and specifications that might help in programming or configuration of specific applications.

There are at least three (3) fundamental approaches to implementing and utilizing HAML instance documents:

a. Application description: Define a haptic system description that might be used in the future to build similar applications – given equivalent requirements/specifications.

b. Feature Description: The HAML description is obtained from the device/API/model via a manual, semi-automatic or automatic extraction. The descriptions are saved in a storage system for later use.

c. Haptic Application Authoring/Composition: In this approach, the system receives a query and finds out a set of descriptions matching the user's query. Then an intelligent agent filters the descriptions to compose the haptic application by performing some programmed actions such as wrapping the API, adapting the haptic rendering algorithms, building the virtual environment, etc.

3.2 HAML Specifications

This section provides insight to the HAML schema and its capabilities to describe multimedia application incorporating audio, video, and haptic media. In particular, the communication description scheme will be presented in details to show how network resources and application requirements are represented using HAML notation. The communication framework (named Admux) we are proposing in this thesis is based on the communication description scheme.

3.2.1 HAML Schema

The HAML structure has been revised since its first version (version 1.0) was proposed in [66]. The latest HAML version (version 2.0) is divided into two generic description schemes: Application Scheme and Environment Scheme. An overview of HAML structure is shown in Figure 3.1. The Application Scheme describes the application requirements and specifications that are static as they do not change during an execution instance of the application. This information should be exchanged ahead of time before starting the application to configure different components of the application (audio-visual and haptic rendering, haptic device, data types, and application meta-data). The Application Scheme comprises several sub-schemes: the Application Meta-data scheme, Haptic Device scheme, Haptic API scheme, Haptic Rendering scheme, Graphic Rendering scheme, Quality of Experience (QoE) scheme, Communication scheme, and Data Type scheme.

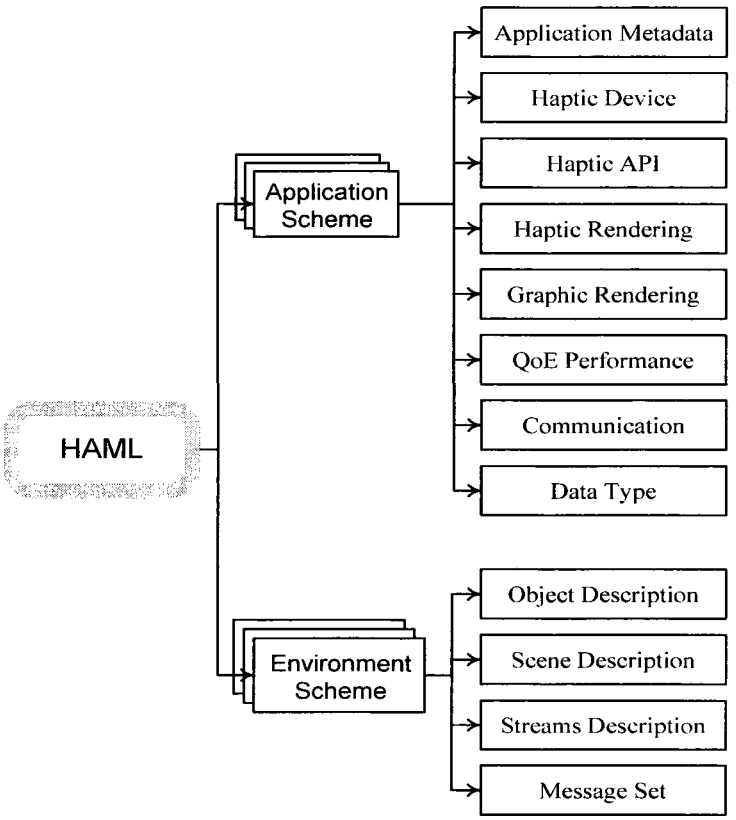

Figure 3.1: HAML schema structure.

On the other hand, the Environment Scheme describes the dynamic features of the Haptic-audio-visual application. That is, the spatial and temporal properties which are specific to a particular execution instance. This scheme is subdivided into four sub-schemes: the Scene Description scheme, the Object Description scheme, the Channel Description scheme, and the Message Set scheme. The Scene Description scheme conveys the spatial and temporal layout of the media objects in a scene. The scene is defined as a composition of one or more objects that facilitates the manipulation of the scene contents (such as adding or deleting objects).

The object description, being uniquely identified by an ID, enumerates the channels (called Elementary Channels) in a communication and specifies how they relate to media objects. The Channel Description scheme includes information about channel ID, channel type, compression scheme, channel priority, media codecs, QoS descriptions, in addition to the synchronization configuration within or among the different channels. Finally, the Message Set scheme defines the communication messages specifications that are exchanged between the communicating parties. These messages are classified as: scene graph messages, object messages, and control messages.

3.2.2 Scene Graph Description

Inspired by the Virtual Reality Modeling Language (VRML), the scene description defines the multimodal environment as a hierarchy or tree of objects. The hierarchical structure enables objects to be grouped semantically, depending on spatial and/or temporal relations. There are a minimum of three types of objects in a scene: visual, auditory, and haptic. The visual component can be a 2D or 3D world. The auditory component can have one or more sources (mono, stereo, etc.). The haptic component comprises the representation of the haptic device (such as the Haptic Interaction Point (HIP)).

A scene consists of an arbitrary number of media objects (visual, audible, haptic, etc.), each represented by a node that abstracts the interfaces to those objects. This allows the manipulation of an object's properties independently from the object media. There are many kinds of nodes that define the structure and properties of the scene including visual nodes (shape, geometry, appearance, material, haptic, etc.), audio nodes (sound, sound2D), interpolation (time sensor), and script or conditional nodes. The most

frequently used nodes are listed in Table 3.1. A scene can dynamically be changed using update commands or by defining an animation mechanism. The basic scene updating commands are: insert, delete, and replace.

Similar to the concept of MPEG-4, nodes are composed of primitive components called fields that act as attributes and interfaces of the nodes. A field has four characteristics: a name, a value, a type of the value (single or multiple values), and a type of behavior (if the field is constant, modifiable, an event source or destination, etc.). A haptic interface is implemented by creating a haptic node. This node contains information related to the haptic rendering of an object, including the stiffness range, static friction range, dynamic friction range, elevation grid range, and texture. The fields have default values that simulate forces that are equal in magnitude and opposite in direction to the forces applied by the user and thus giving the feedback of a solid object. The haptic fields can be assigned different values to create, for instance, a rubbery texture. The haptic rendering algorithm uses these fields to define the physical properties of the objects. Figure 3.2 shows an example of HAML document that describes the scene graph with haptic properties.

3.2.3 Object Description

The object description serves as a link between the scene description and the channels descriptions by aggregating a number of other descriptors. The object description information is usually encapsulated in an Object Descriptor (OD). The OD is uniquely identified by an Object Descriptor Identifier (OD_ID) that allows the scene description to refer to a specific descriptor. The OD provides detailed information about the associated channels, including its location, type (video, audio, graphics, haptics, scene

description channel, object descriptor channel, clock reference channel), channel priority, the intellectual property identification, and an optional extension description. The intellectual property identification provide information that might help to decrypt the elementary channels or that contain authorization or entitlement data. Furthermore, the OD contains semantic information about the object and hooks for content access management.

Table 3.1: Frequently used nodes.

Node Type	Node	Example Fields	Purpose
Graphic	Geometry	Primitive, vertex list, face list, text	Refers/defines graphical primitives or objects
	Appearance	Material 3D, texture 3D, texture transform	Defines the appearance properties of an object
	Material	Color, transparency, intensity, physical properties	Defines physical properties of the composing object/primitive
Haptic	Haptic	Stiffness, static friction, dynamic friction, roughness	Includes the haptic properties of an object
	Navigation	Browsing, targeting, searching, zooming	Defines the interaction paradigm with the environment
Auditory	Audio Source	Source, intensity, speed, effects	Define sound data and access information
	Acoustic	Mono, stereo, reflectivity	Sound type and properties
Grouping	Shape	ID, Node reference, URL, Time, Children, coordinate system, translation, rotation	Groups other nodes defines the rendering order. Each grouping node defines a coordinate space for its children.
	Group		
	Transform		

```
<? xml version="1.0"      ?>
< HAML >
    < ApplicationDS    >... </ ApplicationDS    >
    < AuthorDS    >... </ AuthorDS    >
    < SystemDS    >... </ SystemDS    >
    < SceneDS    >
        < Object    >
            < Type    >... </ Type    >
            < Name    >... </ Name    >
            < Location    >... </ Location    >
            < Rotation    >... </ Rotation    >
            < Geometry    >
                < VertexList    >
                        < Vertex    > 0, 1, 0    </ Vertex    > ...
                </ VertexList    >
                < FaceList    >
                        < Face    > 1, 2, 3    </ Face    > ...
                </ FaceList    >
            </ Geometry    >
            < Appearance    >
                < Material    >... </ Material    >
            </ Appearance    >
            < Tactile    >
                < Stiffness    > 0.8 </ Stiffness    >
                < Damping    > 0.9 </ Damping    >
                < SFriction    > 0.5 </ S Friction    >
                < DFriction    > 0.3 </ DFriction    >
            </ Tactile    >
        </ Object    >
    </ SceneDS    >
</ HAML    >
```

Figure 3.2: An excerpt from a HAML document for the scene graph.

The linking of the object description to the scene is achieved in a two-stage process. First, the object descriptor ID is referenced by the scene description channel and thus links the object descriptor to the scene. The second stage involves the actual binding of elementary channels described by the elementary channel descriptors included in this object descriptor, using the elementary channel ID, which is part of the elementary channel descriptor. This concept is explained in Figure 3.3.

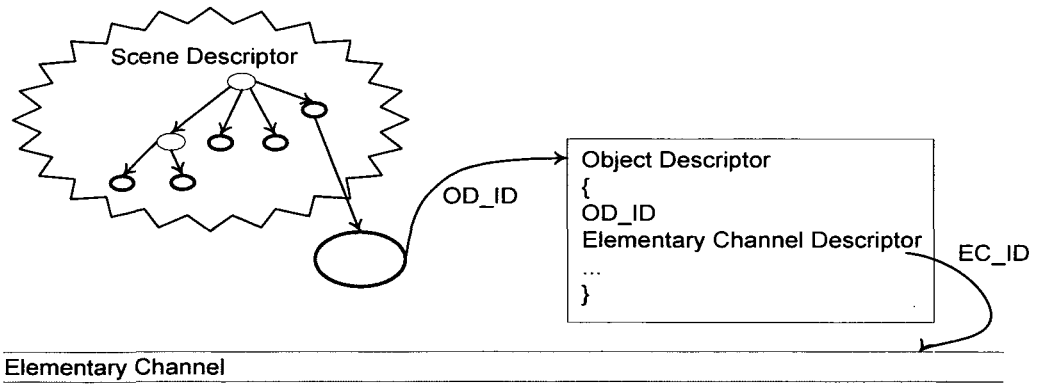

Figure 3.3: Linking the scene graph to the elementary channels.

3.2.4 Elementary Channel Description

An Elementary Channel (EC) carries only one type of data (haptic, graphics, video, or audio) from a single encoder. The EC Descriptor (ECD) is a container for information about the EC. The ECD is uniquely identified by an ID or a URL. The reason for the presence of a URL is to facilitate the description of distributed contents. Furthermore, the ECD contains information regarding the compression scheme, the codec configuration, and the communication scheme.

The codec configuration description contains information needed to initialize and operate a media codec for the elementary channel such as object type, channel type, average and maximum bit rates, as well as the buffering scheme. The synchronization description carries information related to decoding and composition timestamps. The QoS specification description defines the QoS parameters associated with the elementary channel. This includes the maximum allowable end-to-end delay, preferable end-to-end delay, delay variations (jitter), the packet loss rate, the maximum size of packets in bytes, and the maximum arrival rate of packets.

One crucial part of the ECD is the communication scheme. The communication scheme describes the QoS requirements for each media channel. The scheme comprises five different categories of communication specific information: the transport protocol, the synchronization scheme, compression, control method, and multiplexer (as shown in Figure 3.4). The transport protocol defines the transport-layer protocol in use (such as UDP or TCP), the reliability mechanism in use (such as positive or negative acknowledgment), the QoS parameters describing the current network resources, and the network monitoring protocol used to measure the current network QoS (such as ICMP).

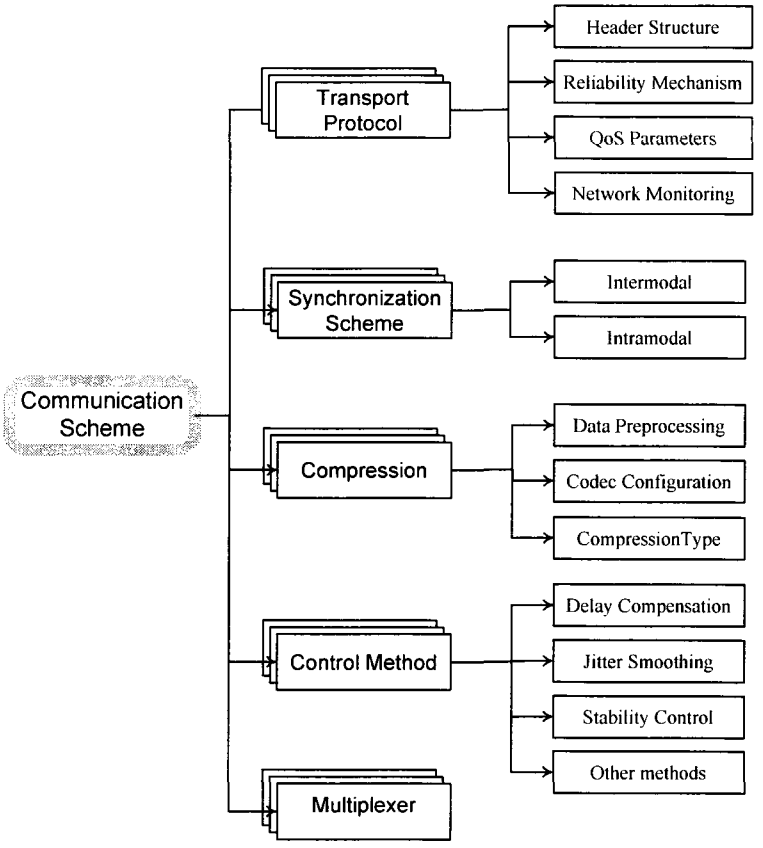

Figure 3.4: Communication scheme.

The synchronization scheme defines both the intermodal and intramodal synchronization models used in the communication. For instance time stamping can be used for intramodal synchronization (for video or haptic data) whereas reference channel can be used for intermodal synchronization. The compression scheme defines the data preprocessing method (if any is used) and the codecs configurations (such as compression rate, speed, etc.) and the compression type (lossy or lossless). The control method scheme describes algorithms that are used at both ends of the network to compensate for network deficiencies. For instance, delay compensation methods minimize the effect of delays and delay variances (jitter) for time-sensitive communication. Similarly, stability algorithms are used to ensure the stability of interaction, particularly the haptic device. Finally the multiplexer scheme stores information about the configuration of the multiplexer including several coefficients that define the performance of the multiplexing scheme. Figure 3.5 shows an excerpt of the communication scheme using HAML notation.

```
<?xml version="1.0" ?>
<CommunicationDS xmlns="http://www.mcrlab.uottawa.ca" xmlns:xsi="http://
www.mcrlab.uottawa.ca/XMLSchema-instance" xsi:schemaLocation="http://
www.mcrlab.uottawa.ca Communicationds.xsd">
   <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>
      <CompressionType>Lossy</CompressionType>
      <Preprocessing>Fast Fourier Transform</Preprocessing>
      <CodecConfiguration>
          <CompressionRate>40</CompressionRate>
          <CompressionTime>20 us</CompressionTime>
          <CompressionQuality>40%</CompressionQuality>
      </CodecConfiguration>
    </Video>

  <Haptic>
      <CompressionType>Lossless</CompressionType>
      <Preprocessing>Low pass filter</Preprocessing>
      <CodecConfiguration>
          <CompressionRate>60</CompressionRate>
          <CompressionTime>5 us</CompressionTime>
          <CompressionQuality>20%</CompressionQuality>
      </CodecConfiguration>
    </Haptic>

  </Compression>
   <ControlMethods>
    <DelayCompensation>Local Lag</DelayCompensation>
    <JitterSmoothing>Timewrap</JitterSmoothing>
    <Stability>Damping</Stability>
   </ControlMethods>

   <Multiplexer>
    <MuxType>Statistical</MuxType>
    <PriorityVector>[1,1,1,1,1]</PriorityVector>
    <WeightVector>[1,1,1,1,1]</WeightVector>
   </Multiplexer>
</CommunicationDS>
```

Figure 3.5: An Excerpt of the Communication scheme.

3.3 HAML Framework

The HAML framework is designed to prove that the HAML description could be utilized to make devices, APIs, and their corresponding rendering/collision algorithms loosely coupled with the application development. In order to accomplish this goal, the haptic component is separated from the virtual environment so that the haptic API could be changed without affecting the environment. The basic components of the HAML framework are shown in Figure 3.6.

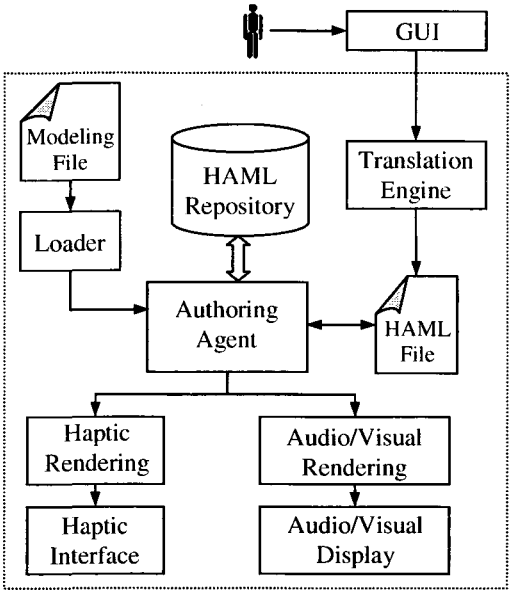

Figure 3.6: HAML framework.

The user interacts with the whole framework via the GUI component that captures the basic user requirements such as the interaction type/device, the virtual environment components, data recording, etc. These requirements are then passed through the translation engine, which relies on the HAML schema to "pump-out" a HAML-formatted document. This document holds a startup/default configuration of the

haptic application required for the framework to work - a discussion of the structuring of HAML will be held later in section 4.3. The Authoring Agent (AA) parses the HAML file to dynamically create the haptic application by selecting and composing components – haptic device, rendering engines, collision detection engines, graphic components, and APIs – that meet the specifications defined in the HAML file and are compatible as well. Notice that the HAML repository stores HAML-formatted description for all available devices, haptic and graphic APIs, and all related information. At this stage, the HAML document is not yet complete, as we are missing the information regarding the newly authored virtual environment application. After the environment has been created and finalized, a "commit" function is performed to update the HAML document accordingly, which means that the HAML document is no longer static.

3.4 Summary

This chapter introduced the design and development of HAML as a foundation description language which is used as a major component in the design of the communication framework. The next chapter details the design, development, of the adaptive communication framework, including the design philosophy, content and access management, and the packetization protocol.

CHAPTER 4

Adaptive Multiplexing Framework for Multimedia Communication

Powered by the five requirements defined in chapter 1, we propose Admux: an adaptive multiplexing framework for multimedia data communication. Admux enables the application to define the communication QoS for multiple media channels based on the application requirements and network conditions. The application requirements are defined using standard HAML syntax [67]. It is worth mentioning that Admux does not include any transport facility; rather it defines an adaptive multiplexing scheme based on the concept of "adaptive media prioritization".

4.1 Design Philosophy

The distinguished features of Admux include the followings: (1) being an application layer protocol, Admux is highly customizable and adaptable to the application requirements and needs via HAML descriptions, (2) Admux utilizes independent channels for different media data that are multiplexed into one transport stream, (3) the protocol is designed to support media prioritization so that proportional resources are assigned to each input channel, (4) and finally channel prioritization can be dynamically changing based on the network conditions and application interactions/events using multiple buffering schemes.

4.1.1 Application Layer Protocol

As shown in Figure 4.1, Admux operates at the application layer – just under HAML that acts as an interface between the communication protocol and the application. Among other advantages, HAML enables applications to configure the communication protocol based on the haptic interface in use and the application requirements. Therefore, haptic devices can be used at any network node without any changes in the application or network configurations. In Figure 4.1, HAV application stands for Haptic Audio Video application.

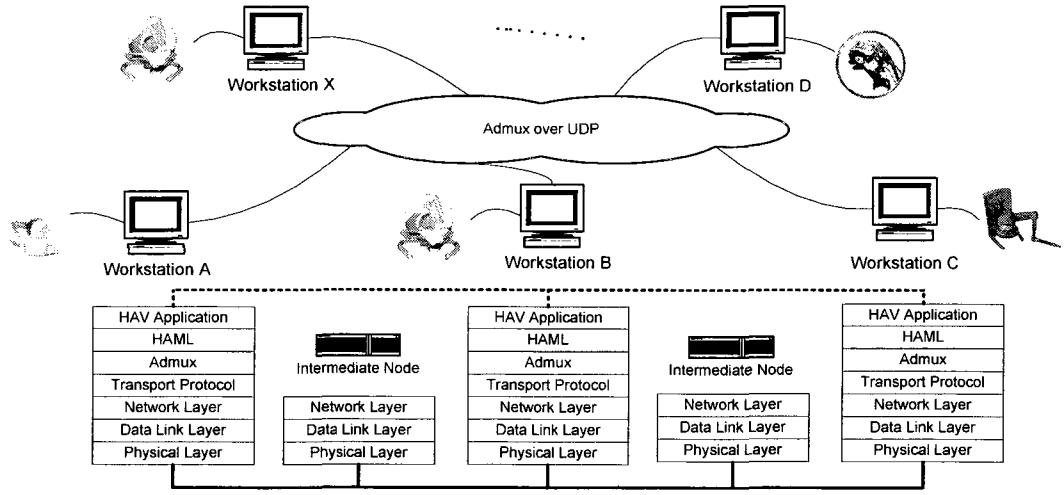

Figure 4.1: Network protocols stack with Admux and HAML.

To further improve the customizability of Admux, the desirable QoS parameters are defined in a HAML description scheme called communication scheme. The communication scheme describes parameters such as the maximum allowable response time, jitter, packet loss, shared objects priorities, security, and trustworthiness and availability. It also provides information about the number of channels used, the

multiplexing scheme configuration, and prioritization rules for the different media channels.

4.1.2 Multiple Media Channels

Admux adopts a flexible delivery mechanism using independent channels for different media data. For instance, haptic data is transmitted over a separate logical channel, as with audio, video, and graphic data. This approach has several advantages. First, the multi-channel feature enables independent sequenced delivery to avoid unnecessary head-of-line blocking among channels. For instance, some applications may only need partial ordering of data units while others might even be satisfied with a reliable transfer without any need for in-sequence delivery. Second, due to the varying network requirements for each media communication (haptic, video, audio media), each channel can be assigned proportional share of network resources to maximize the overall perception of the application. Finally, priorities can be attributed to different channels depending on the important of the communicated data.

Typical applications might have five channels (video, audio, haptic, Graphics, and control channels) that are communicated in a bidirectional manner. It is worth mentioning that each channel has at least two buffers associated with it: transmit (multiple) buffer(s) and a receive buffer. These buffers are introduced to facilitate the communication between the multiplexer and the media channels.

4.1.3 Multiple Buffer Scheme

It is also possible that multiple buffers are used for the same media channel; each buffer is assigned to a particular object data. The multiple buffer approach has two

main advantages. Firstly, it is possible to attribute a priority for each buffer and thus enforce event-based prioritization. The priority of the buffer corresponds to the importance of the object in the environment. Objects with higher priority will get precedence during the transmission of updates. This extends Admux abilities to be adaptable to application specific interactions and events.

Secondly, attributing a retransmission queue for each object makes the protocol more flexible. In the single buffer approach, copies of all the key updates that are transmitted are queued in the same re-transmission queue. If the sending data rate is high enough, the queue will be full of obsolete key updates in the retransmission queue since the application has probably produced newer ones. For graphics data, the newest key update is the most important one, since it holds the new state of the object. On the other hand, discarding an important haptic key update message may result in producing higher than intended forces and thus instabilities in the haptic display. Therefore, for graphics objects the size of the retransmission queue can be set to one whereas for haptic objects the size can be varied according to the object's properties and application needs.

4.1.4 Interaction-based Prioritization

Admux dynamically changes the multiplexing scheme and the compression configuration based on: the network condition and the application events. For instance, if the available network bandwidth decreases, the compression rates increase (consequently the resolution decreases) to maintain stable communication. Admux adapts the multiplexing scheme based on events generated by the application. For example, if the user is in continuous collision with an object, the collision information is highly important and thus Admux gives it even higher priority for transmission. Object-based

compression can also be used at the application level to provide higher resolution representations of objects that define the foreground of the application.

4.2 Admux Communication Framework

This section describes the functional structure of Admux communication framework. An overview of the proposed framework is depicted in Figure 4.2. The application generates multiple streams of media data. First of all, these streams are compressed using different codecs (depending on the media type) and then the compressed streams are multiplexed using the Mux block. Based on the available network resources, the multiplexer dynamically re-configures the codecs to comply with the available resources. Finally, the multiplexed stream is packetized and transmitted using underlying transport protocols (such as UDP or TCP). The same process is performed at the receiver side when the received data is demultiplexed and forwarded to the corresponding destination channels. Admux framework consists of the following building blocks:

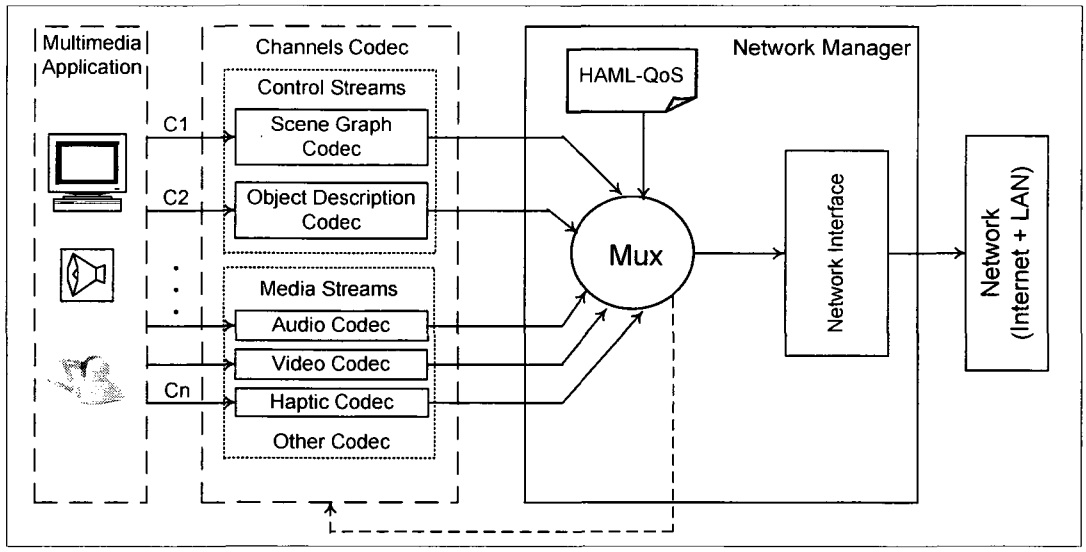

Fig. 4.2: Overview of the communication framework.

- *Channel Codec:* This component defines the coding/decoding mechanism associated with each media communication. For instance, several audio/video codecs have been developed for real-time communication that can be incorporated in this component (MPEG, H.261, H.263, etc). On the other hand, few efforts have been made to develop a haptic codec that optimized the coding/decoding of the haptic data. The multiplexer should be able to dynamically reconfigure the channel codecs to adapt the media streams to the network conditions. Notice that each codec has two buffers associated with it: a transmit buffer and a receive buffer, to facilitate communication with the multiplexer.

- *Multiplexer/Demultiplexer:* As every multiplexer does, this component provides a way to interleave data from different elementary channels into one serialized bitstream. The Admux requires low multiplexing/demultiplexing overhead and minimum processing delays in order to minimize the end-to-end delay of the

application. The configuration coefficients (such as priority vector and weighting coefficients are defined in a HAML file that can be modified at run-time.

- *Network Interface:* The network interface provides the transport layer mechanism for Admux communication. In the current implementation of Admux, the Network Interface uses UDP. UDP is chosen for its simplicity and speed. It does not impose any reliability schemes that are not usually needed in real time applications. Notice that reliability schemes can be defined in this case at the application layer, or can be implemented within the Admux framework.
- HAML-*QoS:* This is an instance of the Communication scheme as specified by the HAML description. It defines the transport and multiplexing mechanisms of the multimodal information such as the quality of service parameters per each input channel, the default values for the coefficients used to (re)configure the multiplexer, etc. It also contains the number of input channels and their respective network requirements and the associated codecs configuration.

4.3 Content Access Management

A major task of the Admux architecture is to describe the relations between media components (objects) of a scene. The description is performed at two levels: structure level and contents level. On the structural level, the scene description defines how the media objects are arranged in space and time. On the contents level, the object description describes how the various channels that contain the media data relate to each other and how they are configured and synchronized.

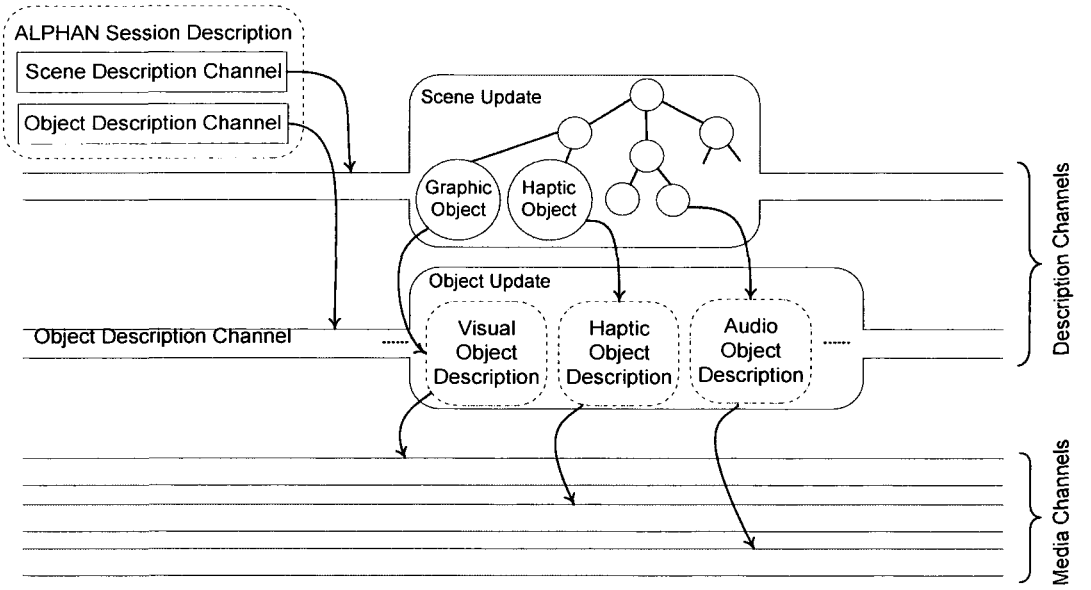

Fig 4.3: Content access management.

The content access procedure always starts with a session description (Figure 4.3). The session description points to at least two essential channels: the scene description and the object description channels. The session information is exchanged between the communicating parties using an initial communication path via signaling protocols such as RTSP [68]. Notice that the contents and control descriptions are separated to ease content management. The communicating entities, then, establish the two channels (scene graph channel and object description channel). Then they exchange the scene graph description through the scene description channel along with update messages to add their own haptic device object to the scene graph at each end. The configuration might change if a client-server architecture is used for the communication.

Admux protocol defines several types of messages such as scene and graph messages, session messages, and media channel messages. For instance, the

`InitialSession` message is used to initiate the communication session by defining the scene description and object description channels. After agreeing on these two channels, the communicating parties initiate the two channels and start communication the scene graph the object description information. One party sends a `SceneUpdate` message with its graph nodes to the other party (including the haptic interface object represented by a haptic node) whereas the other party integrates the received nodes into its own scene graph and sends back a `SceneUpdate` message with the total scene. The first party updates its local scene graph accordingly. The media channels (elementary channels) will be instantiated based on the object descriptions and media communication starts. A summary of the most common Admux messages are shown as a sequence diagram in Figure 4.4.

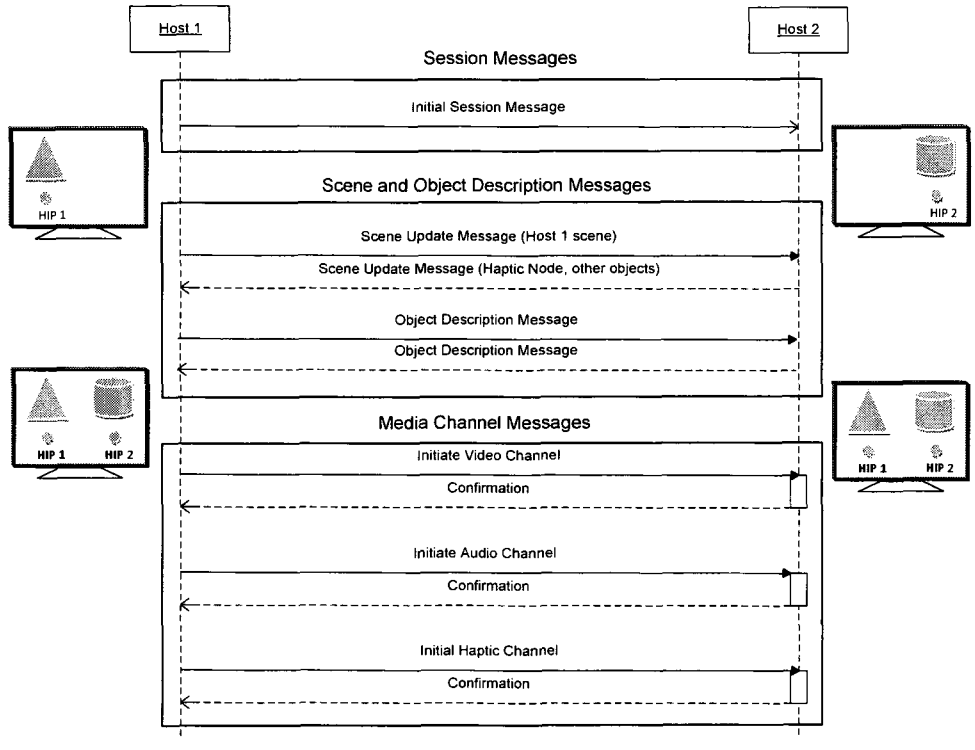

Fig 4.4: Different types of Admux messages (HIP: Haptic Interaction Point).

4.4 Admux Packetization

Admux packetization of media streams is done in two steps: first the Packetized Elementary Stream (PES) packets are composed from the elementary stream data and then PES packets are encapsulated in Transport Stream (TS) packets. The PES packet stores one update/frame of a media stream and thus its size is media dependent (varying). The TS packetization is designed to allow multiplexing of equal size data units and to synchronize the output. Once the TS packet header is added, the TS packets are stored in the channel transmit buffer(s) for multiplexing. Typically, a PES packet may be much larger than a TS packet. Figure 4.5 shows the packetization process as a two-phase process.

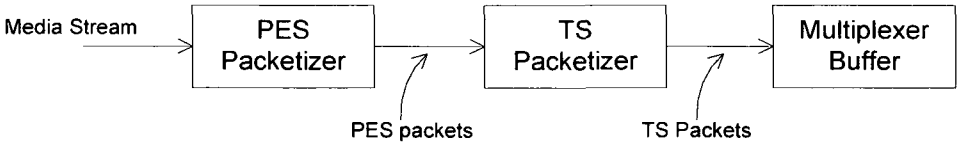

Figure 4.5 The packetization process.

All the header fields are carried in the network byte order, that is, most significant byte first. This byte order is commonly known as big-endian. Padding bytes are filled with zero. The timestamps of Admux are 32 bit fields. Participants of an Admux simulation should synchronize their clocks, so that they can correctly identify the current timestamp cycle.

4.4.1 PES Packet Structure

The PES header carries various rate, timing, and descriptive information as set by the encoder. The PES packet length is described in a field provided for that purpose.

The PES packetization interval is application dependent resulting in packets of variable length with a maximum definable size of 216 bytes. The PES packet header is shown in Figure 4.6. The followings describe each field briefly.

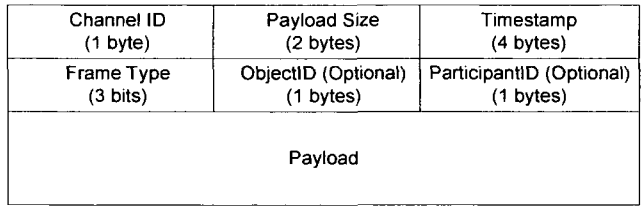

Figure 4.6: The PES packet header.

- Channel_ID: Each elementary channel is identified by a unique Channel_ID which is carried by the PES packet. This ID is also used by the demultiplexer to identify to which destination channel the data must be forwarded to.
- PayloadSize: This field serves to specify the exact size of the PES payload. This field is necessary since each media type has different sizes for their updates/frame units.
- Timpestamp: The timestamp value is expressed as milliseconds that have passed since 0h UTC on 1 January 1900 modulo 232. The timestamp indicates the instant of time where the update must be consumed at the receiver side and is used for synchronization. The field might also be used to establish global ordering among all updates generated by all the participants. Such global ordering is very useful when it comes to implementing consistency assurance mechanisms like timewrap.
- Frame Type: Three bits are reserved to specify the type of the update. This means that up to eight update types can be defined. Only four types are predefined (see

Table 4.1). Up to four other types of updates can be defined by application developers in case they contrive other types that better suit their application requirements.

- ObjectID: Every update is sent to uniquely modify the state of one object in the environment. The Object Identifier (ObjectID) uniquely identifies an object in the environment. Eight bits are reserved for this field, which means that an environment can hold up to 256 objects. An ObjectID is persistent for the lifetime of the simulation. This field is optional.
- ParticipantID: The Participant Identifier (ParticipantID) uniquely identifies a participant. This participant is the original sender of the packet. Admux does not define a specific mechanism to choose the ParticipantIDs. A ParticipantID is persistent for the lifetime of the simulation. If an application requires multiple Admux sessions, the ParticipantID remains the same for the participant across all of these sessions. This field is of particular importance in case of collaborative environment and thus is optional.

Table 4.1: Predefined update types.

Update Type	**Description**
Key Update	Holds data representing key events that must be conveyed to the other side reliably. Such events are defined by the application that runs on top of Admux. Note that collision updates are sent reliably regardless of the update type.
Normal Update	Makes up the majority of the updates that are sent over the network. They are sent unreliably over the network.
Incremental Update	Useful when bandwidth is limited, these updates hold incremental information with respect to the last received key update. These updates are especially useful when key updates are being often sent; this would produce small size incremental updates.
Control Update	Holds no useful information in its payload. The control update is used for the delay and jitter estimation between two hosts on the network.

4.4.2 TS Packet Structure

Each PES packet is fragmented into fixed-sized transport packets (TS packets) to form a general purpose way of combining several channels, possibly with independent time bases. This is advantageous when there is a need to send multiple media channels at a time or/and there may be potential packet loss or corruption by noise. The header structure of a TS packet is shown in Figure 4.7. Here is a brief explanation of each field.

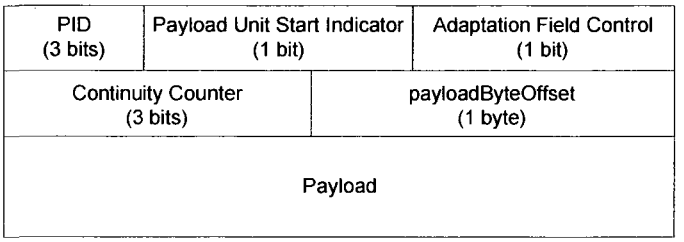

Figure 4.7 The TS packet header structure.

- Packet Identifier (PID): The PID is used to uniquely identify the channel to which the packet belongs. The PID allows the receiver to differentiate the channel to which each received packet belongs. Some PID values are predefined and are used to indicate various channels of control information. A packet with an unknown PID, or one with a PID which is not required by the receiver, is silently discarded.
- PayloadUnitStartIndicator: This field is a flag that indicates the start of a PES packet payload or the continuation of the previous payload for a PES packet. It is used during the reassembly of PES packets to indicate the arrival of the last TS packet of a PES packet.

- AdaptationFieldControl: This field is a 1 byte length that represents the number of bytes in the adaptation field immediately following this byte. The adaptation field contains additional optional transport fields that can be used for implementation application layer reliability mechanisms. This is crucial since Admux is based on UDP which does not provide any error detection/correction mechanisms.
- ContinuityCounter: This field is a 4-bit counter, which usually increments with each subsequent packet of a frame, and can be used to detect missing packets.
- PayloadByteOffset: The byte offset value of the start of the payload or the length of adaptation field is mentioned here.

4.5 Summary

This chapter presented the details of Admux design and development as per the design philosophy, the framework design and development, data and access management, and communication protocol design. The next chapter introduces the core work of this thesis; the design, analysis, and simulation of the adaptive multiplexing scheme for multimedia data including haptics, audio, video, and graphic data.

CHAPTER 5

Multiplexing Scheme: Mathematical Modeling and Simulation

In this chapter, we present a statistical mathematical model for adaptive multiplexing based on the network conditions and application requirements. The concept is partially derived from competitive learning in neural networks where a number of input channels (represented as neurons) compete for resources and the 'winner' neuron will be chosen by the multiplexer. We also demonstrate the validity of the model by performing a series of experimental simulations and analyze the model to calibrate several coefficients that define the model behavior.

5.1 Mathematical Model

5.1.1 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 Equation 1), and the Current Selection Probability (CSP) vector (W) that represents the currently allocated resources for each channel, and is shown in Equation 2 (N is the number of media channels). The objective is to match the two vectors so that each channel is provided the desired proportion of the network resources.

$$X = [x_1, x_2 \ldots, x_N]^T \quad (1)$$

$$W = [w_1, w_2, \ldots, w_N]^T \quad (2)$$

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 Equation 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 is the chance of a channel to win a resource and thus the channel is statistically given higher portion of network resources.

On the other hand, the CSP vector is dynamic and is computed according to Equation 4. The initial value of the CSP vector (at t=0) is computed as the normalized fractions of the initial network resources and in proportion to the DSP values. At a general instance of time (t>0), the CSP is updated based on two factors (shown in Equation 4, t>0). First, the weighted difference $(x_j - w_j(t))$ 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 $(\mathrm{x_j} - \mathrm{w_j(t)})$ results in slower convergence of the CSP value to its DSP counterpart. This is desired since 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 $\Delta \mathrm{Q} \frac{\mathrm{x_j}}{\sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{x_k}}$ term. The updates in network resources are distributed proportional to the DSP weights.

$$x_i = \sum_{j=1}^{M} \alpha_j * A_{p_j} \qquad \forall i \tag{3}$$

$$w_i(t) \begin{cases} \frac{x_i \sum_{k=1}^{M} \beta_k * N_{p_k}}{\sum_{i=1}^{N} x_i} & t = 0 \\ w_j(t) + \alpha_j \left(x_j - w_j(t-1)\right) Z_j + \Delta Q \frac{x_j}{\sum_{k=1}^{N} x_k} & t > 0 \end{cases} \tag{4}$$

Where:

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

M is the number of QoS parameters

N is the number of input channels

x_i is the desired selection probability of the i^{th} channel.

w_i is the current selection probability based on the network conditions.

α_j is the convergence rate of the j^{th} input QoS parameter

A_{p_j} is the desired value of the j^{th} input QoS parameter

β_k is the weighting factor for the k^{th} network parameter

N_{p_k} is the value assigned to the k^{th} network parameter

a_j is the rate at which w_j converges to the x_j value.

ΔQ is the change in the network conditions

Additionally, the proposed model defines an absolute minimum DSP vector ($X_{(\mathrm{min})}$) that represents the non tolerable minimum QoS requirements corresponding to the just perceivable quality of the media channels (Equation 5). If the minimum requirement of any channel is violated, the multiplexer drops that channel where 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.

$$X_{(min)} = [x_{1_{(min)}}, x_{2_{(min)}} \dots, x_{N_{(min)}}]^T \quad (5)$$

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. The idea of matching the DSP vector to the CSP vector is demonstrated in Figure 5.1.

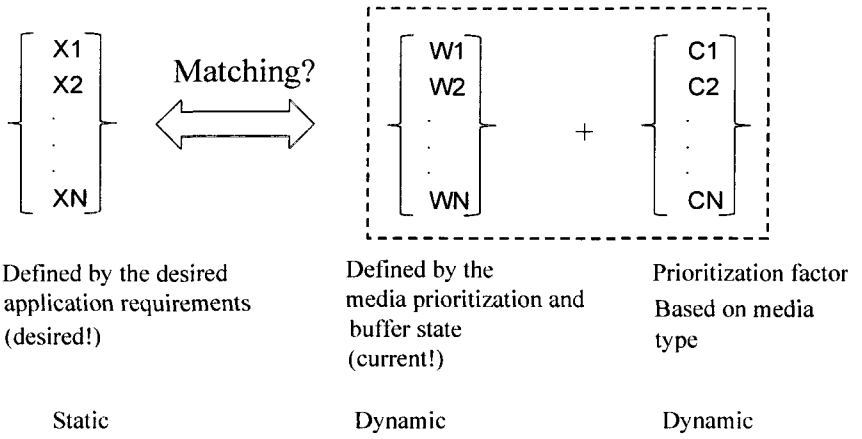

Figure 5.1: Matching the desired quality vector to available resources vector.

5.1.2 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 according to the following steps:

- *Step 1:* The prioritization factor (C_j) is computed as defined in Equation 6. C_j is the summation of 1/N (the equal share of weights), the contribution weight ($f_j(k)$), and the buffer weight ($q_{l_j}.\varphi_j$). The contribution weight $f_j(k)$ 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_j}), as shown in Equation 7. The buffer weight is computed, according to Equation 8, as the summation of the number of data units in the buffer (P_{size_j}) multiplied by the weight coefficient (ρ_j) and the rate of

data units flow (P_{rate_j}) multiplied by its respective weight coefficient (σ_j). The linear prioritization coefficient (q_{l_j}) enables controlling the prioritization factor in a linear manner which may be desired for some media channels.

$$C_j = g(\frac{1}{N} + f_j(k) + q_{l_j}.\varphi_j) \tag{6}$$

$$f_j(k) = 1 - (1 - \frac{1}{N})^{\frac{k_j}{K_{d_j} q_{e_j}}} \tag{7}$$

$$\varphi_j = \rho_j P_{size_j} + \sigma_j P_{rate_j} \tag{8}$$

Where:

g is the fairness gain constant, empirically set to 10.

$f_j(k)$ is the contribution of the j^{th} input channel.

k_j is the TS packet size for the winner channel.

K_{d_j} is the number of winssss for the j^{th} channel.

q_{e_j} is the exponential prioritization coefficient.

φ_j is the state of the transmit buffer of the j^{th} channel.

q_{l_j} is the linear prioritization coefficient.

- *Step 2:* Calculate the intensity $I_j(t)$ for each input channel (which represents the Euclidean distance between w_j and x_j minus the prioritization factor), as shown in Equation 9. The channel with minimal intensity wins the competition and thus is selected by the multiplexer. Notice that $\mathrm{D}\left(\mathrm{w_j(t)}, \mathrm{x_j(t)}\right)$ is a negative value since w_j is always smaller than x_j. Therefore, the higher the $\mathrm{C_j(t)}$, the smaller the intensity becomes (larger negative number) and thus the more likely the channel is to be selected.

$$I_j(t) = D\left(w_j(t), x_j(t)\right) - C_j(t) \quad (9)$$

Where:

$D(w_j, x_j)$ is the Euclidean distance between w_j and x_j.

$C_j(t)$ is called the prioritization factor. It is used to enforce media prioritization.

- *Step 3:* Read the current network conditions and update the CSP values for all the input channels according to Equation 4 (t>0). Notice that the w_j weight will increase for the winner channel whereas the w_j 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_j) controls how fast the CSP values converge to the DSP values.

5.1.3 Model Analysis

The proposed model provides the mathematical foundations for an adaptive multiplexing scheme. First, the available resources are divided among the input channels (in proportion to the DSP values). However, once the intensities for all channels are computed and the winner channel is identified, the corresponding w_j will increase whereas other w_i values decrease. This gives other channels higher chance to win the competition the next time. This continues until another channel is having the minimum intensity, which makes it win the competition and have its w_j value increasing, and so on. Hence the selection of particular input is made to minimize the differences between X and W vectors.

The prioritization mechanism works as follows: as C_j increases, the intensity of the input decreases which gives the corresponding channel a better chance to win the competition. Also, it is worth mentioning that C_j changes linearly with the state of the input buffers and exponentially with the contribution factor ($f_j(k)$). The two coefficients q_{l_j} and q_{e_j} can be adjusted at run-time to dynamically change the prioritization behavior of the input channels. This is very useful in case a media data becomes more important than the others at a particular instance or during certain events/conditions. For instance, the haptic device position becomes highly important in case the user's avatar is colliding with the virtual world or grabbing an object. If the user is exploring the free space, the haptic data become less important than, for example, audio data. In case $q_{l_j} = q_{e_j} = 1$, no dynamic control of prioritization is enforced.

The exponential prioritization factor (q_{e_j}), as shown in Figure 5.2, changes the prioritization factor in an exponential manner. This implies that if each channel is assigned different q_{e_j} then the respective contributions can be made exponentially changing. This provides the designer with higher flexibility in the prioritization control, depending on the application requirements and perceptual needs. Furthermore, the winning weight has an exponential impact on the prioritization factor (shown in Figure 5.3).

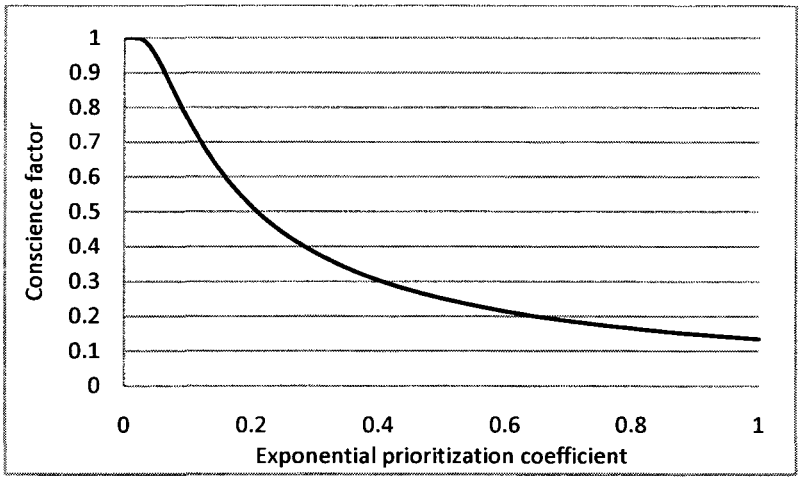

Figure 5.2: Effects of q_{e_j} on the prioritization factor.

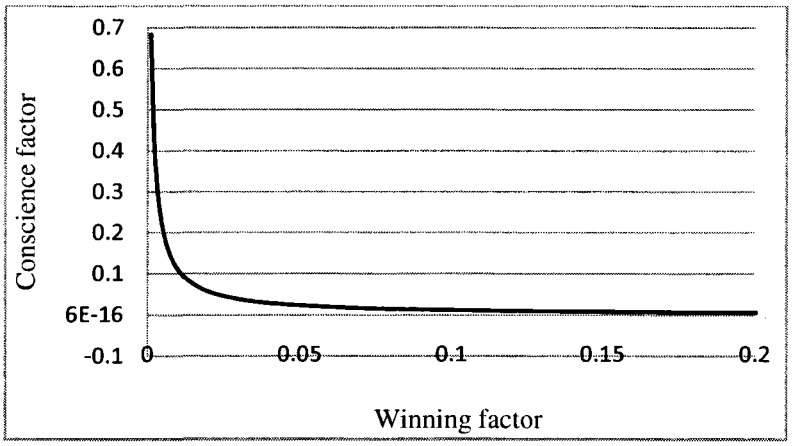

Figure 5.3: Effects of winning factor on the prioritization factor.

The selection of the values for the various coefficients used in the multiplexer model has a direct impact on the performance of the multiplexer. Therefore, we performed empirical studies and found that the DSP and CSP vectors (X and W) should be normalized (between 0 and 1), g = 10, a_j=0.2, $q_{e_j} = 0.5$, $k_j = 4$, $\rho_j = 10^{-4}$, $\sigma_j = 5 \times 10^{-3}$. These values have been calibrated with four media channels: audio, video, haptic, and graphic channels, and resulted in the intended performance of the multiplexer. The

intended performance is demonstrated when the w_j values for all channels converges to the desired x_j almost at the same time and at rates proportional to their respective prioritization values.

5.2 Admux Software Implementation

Admux framework was implemented and developed in C++ using the Microsoft Visual Studio 2005. The implementation library consists of four primary classes and three auxiliary classes. As shown in Figure 5.4, the primary classes are `StreamManager`, `Admux`, `Network`, and `Demux`. The auxiliary classes are: `Buffer`, `Packetizer`, and `Transmitter`.

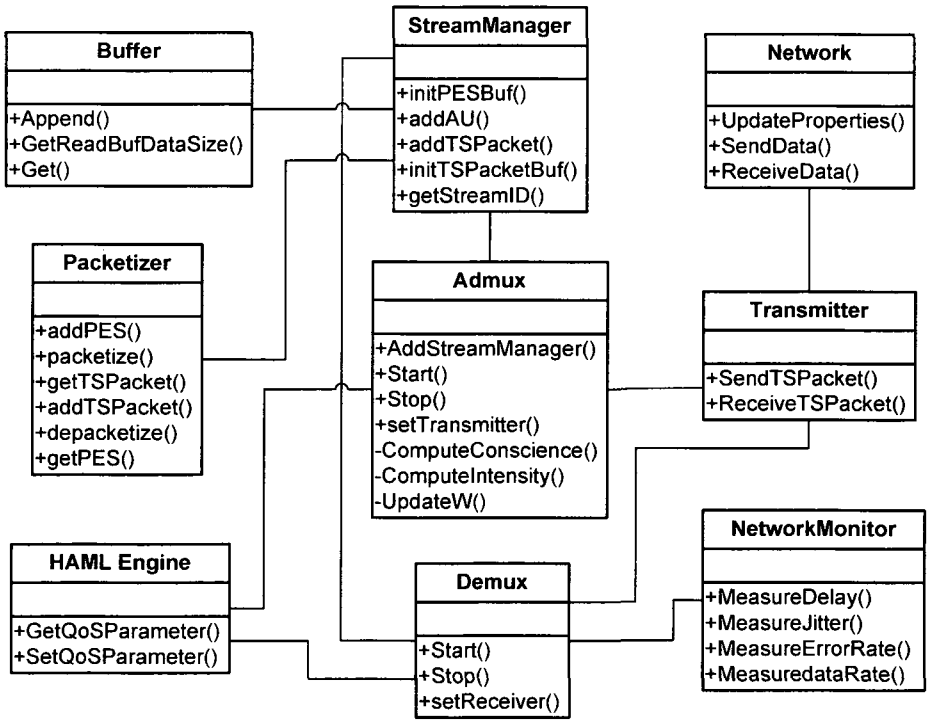

Figure 5.4: UML class diagram for Admux simulation.

5.2.1 Admux Library

Here is a brief explanation of the classes that makeup the Admux library, along with explanations on how they interact with each other:

`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.

`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()`.

`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 `GetReadBufDataSize()` method), and retrieve the buffer information (using the `Get()` method).

`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 to as Packetized Elementary Stream packet (PES packet),

which might have variable length. Therefore, the PES packets are fragmented into smaller fixed sized data units called Transport Stream packet (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, then forward the PES packet to the receive buffer.

`Transmitter`: This class provides the `Admux` class an interface to the transport protocol. It handles communicating the TS packets into the network using (in the current implementation) 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).

`Demux`: This class implements the demultiplexing of the multi-modal stream into the destination channels. The received data is passed to the `Demux` class via the `Transmitter` class, get 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()`.

`Network`: This class simulates the network conditions. The simulated QoS parameters are the network delay, jitter, data rate, and a 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 return the data using the `ReceiveData()` method. The network properties can also be updated using the `UpdateProperties()` method.

`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 desired QoS parameters using the `SetQoSParameter()` method.

`NetworkMonitor`: This class measures the network QoS parameters as per each communication channel. In the current implementation, this class implements computing the average delays, jitters, 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()`, `measureErrorRate()`, and `MeasureDatarate()`, each returning an array of values for the respective quality parameter. For instance, `MeasureDelays()` returns an array containing the average delays for the four media channels as computed at the de-multiplexer side.

5.2.2 Admux Library Stability Analysis

As a communication framework, it is important to assess the stability of Admux framework. The approach adopted in our analysis (proposed in [69]) 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 is seldom used – which is our case. Our results show that DSC is 24 whereas NSI is 21, and thus NSI/DSC ratio is 0.875. This shows that Admux architecture is thus stable.

5.3 Model Simulation and Results

In this section, we simulate four media channels with equal frame rates and frame sizes for all the channels (a video object called V object, a haptic object called H object, an audio object called A object, and a graphics object called G object). The simulation traffic is composed of 1000 PES packets per each input media channel and all the results are averaged over the 1000 samples. Finally, a Pentium 4 PC with 2 GB of RAM and 100 Mbs Ethernet cards was used for the simulation.

Six key features of the proposed design are simulated and evaluated in this section: (1) the outperformance of the proposed multiplexing scheme over the use of parallel communication channels, (2) the adaptability of multiplexing to changes in the network conditions, (3) the time complexity of the multiplexer model, (4) the adaptability to application events/interactions and multiple buffering scheme, (5) the multiplexer scalability as per the number of input channels, and finally (6) finding the optimal size of the TS packet.

5.3.1 Multiplexing Scheme Versus parallel Communication

The objective of this test is to prove the effectiveness of the multiplexing approach as compared to parallel communication. Several multi-modal applications use parallel UDP channels for each media data. Notice that in this case, the multiplexing is performed by the network layer which implies that the application has no control over enforcing a multiplexing scheme based on the application and interaction requirements.

We have implemented four parallel UDP channels and compared the delay values per each channel to the case when the multiplexer is used. As shown in Table 5.1, the average delays/jitters are almost the same for all the input channels in case of parallel

channels. This is not always desirable since different media have different QoS requirements. On the other hand, using the proposed multiplexing scheme has resulted in proportional allocation of network resources (delays/jitters as shown in Table 5.1) based on the prioritization associated with each input channel. Therefore Admux meets requirement 3 as it shows that using the multiplexer is advantageous.

Table 5.1: Comparing delays/jitters for parallel and multiplexed channels.

	Approach	**H object**	**A object**	**G object**	**V object**
Delay (ms)	Multiplexer	312.34	359.14	408.82	455.62
	Parallel	433.52	432.68	423.96	444.50
	Desirable	350	400	450	500
Jitter (ms)	Multiplexer	19.75	22.42	31.97	52.70
	Parallel	39.53	40.43	34.25	44.68
	Desirable	25	30	40	60

5.3.2 Adaptability to Application Requirements and Network Conditions

This test evaluates the ability of the multiplexer to adapt and distribute resources in proportion to the DSP values when the network has enough resources for all the media channels. The resources allocation is optimized as Figure 5.5 shows how the weighting factors (w_j) are converging to the desired values (x_j) over time. Notice that the w_j represents network delays which means that the delay associated with each media channel is statistically satisfied. In other words resources are redistributed based on the priority level of the respective media and network resources.

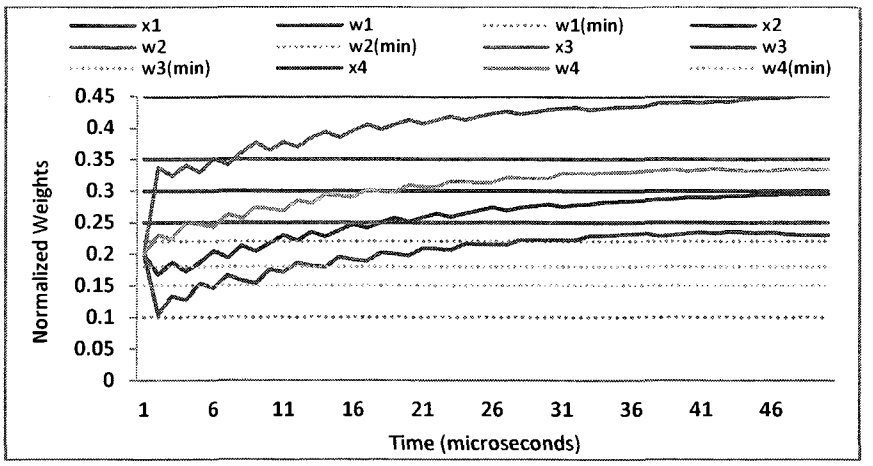

Figure 5.5: Simulation with 4 channels (adaptability of the multiplexer).

Furthermore, Figure 5.5 shows that the absolute minimum thresholds ($w_j(min)$) were never crossed through all the multiplexer operation (which represents the absolute minimum delay that should not be violated). Theoretically, this cannot be always guaranteed since the multiplexer is statistical. This feature is of a significant importance to haptic media communication since a minimum bandwidth is required for stable haptic communication and must not be compromised. This implies that delay compensation and jitter smoothing algorithms should complement the operation of Admux.

Another simulation we performed was to evaluate the stability of the mathematical model, in particular, the convergence of the w_j, based on different network conditions. We simulated three different Internet conditions: (1) delay = 30ms, jitter 5ms, (2) delay = 90ms, jitter 10ms, (3), delay = 200ms, jitter 20ms. As shown in Figure 6, in the three conditions, the w_j values are converging to the x_j values. This implies that Admux adapts to different network conditions and has a stable mathematical model. Therefore, Admux meets requirement 5.

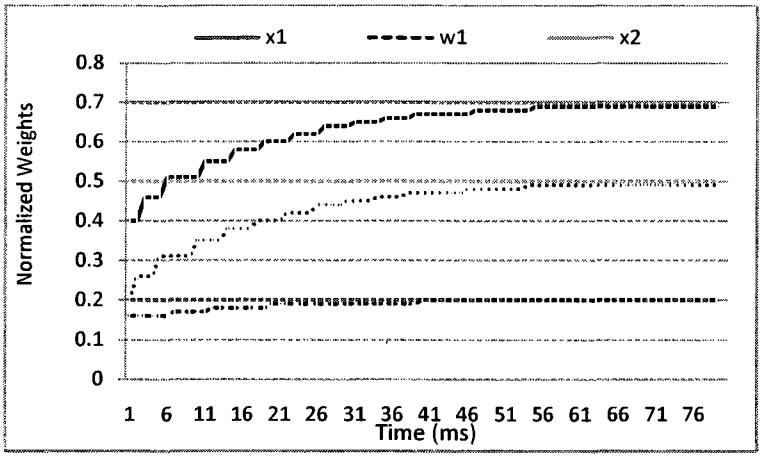

Figure 5.6: Adaptability to changes in network delay.

5.3.3 Time Complexity Analysis

One of the critical factors to examine is the computation time of the multiplexer. This is of particular importance for haptic data communication where computation delays must not exceed 1 millisecond (to enable a 1 kHz haptic rendering loop). Notice that haptic rendering loop is only 1 ms and thus the computation time should be way below. We measured the computation time for each multiplexing cycle and found that the computation time converges to 1.12 μs, as shown in Figure 5.7. This result is comfortably within the 1 ms interval necessary for haptic rendering. Therefore, the multiplexer does not cause significant overhead on the overall computation delay in the software system.

5.3.4 Application Adaptability and Multiple Buffering Scheme

The goal of this simulation is to prove the effectiveness of the multiple buffering scheme over the single buffering approach. To do so, we have simulated an environment containing three objects: Haptic object (H object), Graphic object (G object), and Video

object (V object). All the media objects generate data at the same frame rate and same frame size.

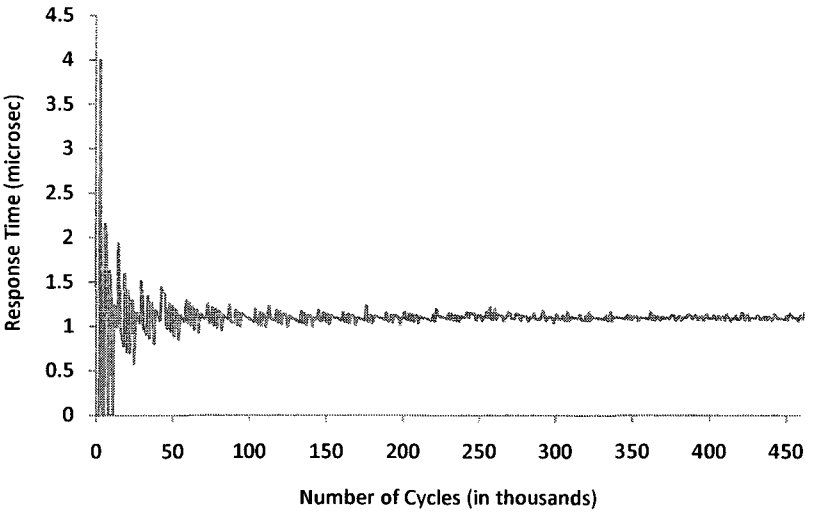

Figure 5.7: Time complexity analysis.

A key advantage of multiple buffering over single buffering is the fact that we can attribute priorities to objects in order to give precedence during transmission for higher priority objects over lower priority ones. We run the simulation first by excluded the G object. Also, it is assumed that the H object is more important to the application than the V object. In the first run, the priorities of Objects V and H were kept the same. In the second run, object V's priority is decreased. We measure the total number of updates received for each object with respect to time. Duplicate updates are not counted. Figure 5.8 shows the result of the simulation performed under ideal network conditions (no delays/jitters). Object V's updates are delayed in favor of object H's updates. In the single buffering approach, the precedence in packet transmission cannot be given to any of the objects as both object's updates are located in the same sending buffer.

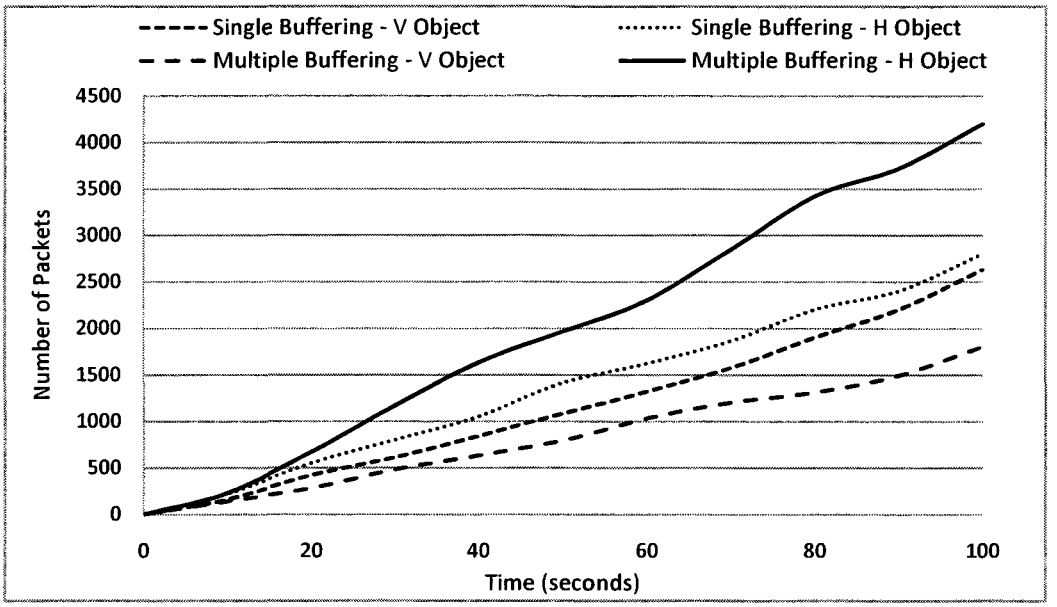

Figure 5.8: Packet throughput with Single Buffering and Multiple Buffering.

Another simulation we conducted was to examine the effects of delays/jitters and errors in the network on the packet throughput with only three objects (H object, V object, and G object). Figure 5.9 shows the results of these simulations. As we start injecting delay, jitter and packet loss into the simulation, the resulting key updates throughput for both approaches (single and multiple buffers) starts to diverge especially for the H object (Figure 5.9). 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 object they belong to. Transferring obsolete updates steals the available bandwidth from the fresher ones, especially 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. These obsolete key updates are sometimes useless even if they make it in the second retransmission and are often discarded by the receiver in favor of newer ones. Also, if we store numerous old updates while maintaining a timer for each, the performance of the application might suffer. The only way to limit the size of the retransmission queue is by stopping to produce newer key updates in order to service the old ones stored in the retransmission buffer, this only can be done if we block the sending buffer, which might further aggravate the performance problem. Moreover, as the retransmission buffer grows bigger, it starts to compete for bandwidth with the sending buffer in order to reduce its size. Therefore, Admux meets requirement 1.

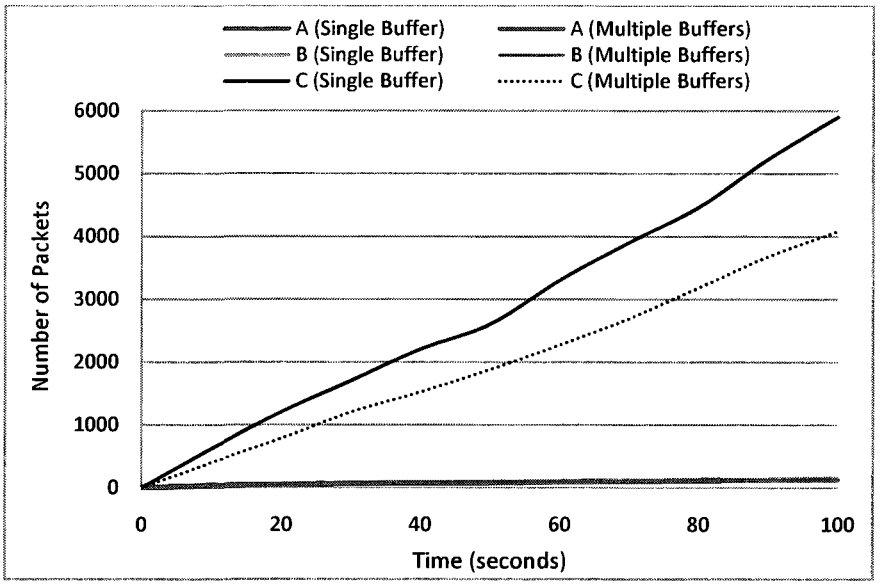

Figure 5.9: Packet throughputs with delay 80ms, jitter 20ms and Loss 2%.

5.3.5 Media Channels Scalability

The scalability test is performed to examine the effects 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. As shown in Figure 5.10, the average computation time is maintained almost constant at around 1.18 μs. Therefore the multiplexer performance is not affected by the increase in the number of input channels (simulated with up to 100 channels, as shown in Figure 5.10).

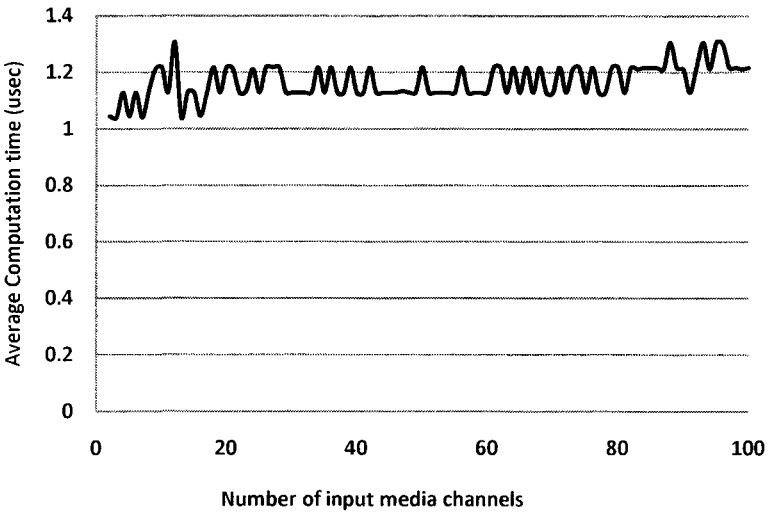

Figure 5.10: Scalability of input media channels.

5.4 Application Simulation and Results

This section presents conduct a simulation study for a real application to evaluate the performance of Admux with realistic multi-modal data. It is worth mentioning that we are simulating only the media elementary channels. The scene graph channel and the object description channels are not included in the simulation since it is assumed that the scene graph and object descriptors are communicated at the beginning of the communication session.

5.4.1 Application Scenario

Figure 5.11 shows an interpersonal communication application, named HugMe system, that we have simulated as a proof-of-concept application for Admux communication framework. As shown in Figure 5.11, the remote person is wearing a haptic suite (haptic jacket) that is capable of simulating nurture touching. The local person uses a haptic device to communicate his feelings with the remote person. A depth camera (will be detailed in chapter 6) 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.

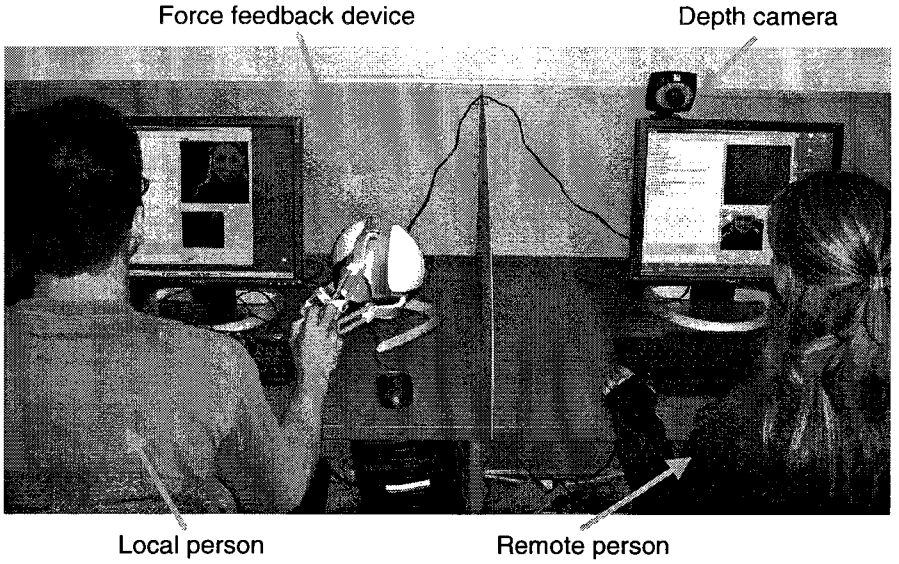

Figure 5.11: The simulated scenario of HugMe system.

The simulation comprises four media channels: the video object (V-object) capturing the remote person, the haptic object (H-object) representing the haptic data

received from the haptic device and sent over the network, the graphics object (G-object) that provides information about the remote person's avatar and any other graphics cues, and the audio object (A-object) that conveys the verbal communication to the remote person.

5.4.2 Simulation Implementation

The simulation implementation is the same used in section 5.2 (Figure 5.4) with one additional class (`TrafficGenerator`) that is responsible for simulating the input media traffic. This class generates four media streams: haptic, audio, video, and graphics with particular traffic characteristics for each. The video object is simulated as 30 frames per second stream of video frames each sized 640x480 pixels (with 24 bits per pixel, we assume a frame size of 10 Kbytes). The haptic object is simulated with update rate of 1 kHz frame rate with a haptic frame size of 32 bytes (position and timestamp information). The audio object is simulated at a 64 Kbps rate with a frame size of 417 bytes (stereophonic). Finally, the graphics object is kept at 30 frames per second and the frame size is assumed to be 100 bytes.

5.4.3 Simulation Results

Similar to the analysis presented in section 5.2, we performed the same set of five simulations to test the performance of Admux but in this case with real multimedia traffic. By simulation, we refer here to simulating the media traffic, in other words, the application itself has been simulated. Furthermore, the network conditions are also simulated. We used a network simulator with 30 ms delay and 10 ms jitter. Also, all the

results are based on simulating communication traffic with 1000 frames per each input channel and all the results are statistical averages over 1000 samples.

5.4.3.1 Parallel versus Multiplexed Communication

The objective of this test is to compare the performance of Admux with real multi-modal traffic for the four media objects (H object, V object, G object, and A object). We have simulated four channels for communicating the four objects with their respective communication properties (frame size, frame rate) using four UDP communication channels. We compared the delay and jitter values per each channel to the case where the multiplexer is used. In this case, the network simulator is set with 30 ms delay and 10 ms jitter.

As shown in Table 5.2, it is clear that the average delays and jitters are almost equal in the case of parallel channels. This is because the application cannot enforce any prioritization for the QoS resources allocation. On the other hand, using Admux has enables the application to assign network resources in proportion to the media needs and application requirements. In this case, all the delays of the four media are met whereas with parallel channels, the delay and jitter requirements were not met for the haptic media. This shows that Admux can adapt to the application needs, particularly when the QoS requirements for each media channel is very different from other media channels.

One implication of statistical multiplexing is that an absolute delay is not guaranteed (all the results are average values). However, haptic modality does not tolerate delays as they result in unstable haptic rendering. Therefore, we investigated – for the haptic modality – the frequency at which the absolute delay was not met (shown in Figure 5.12). Our results show that the percentage of crossing the absolute delay

requirement for haptic modality was 3.16%. This means that 3.16% of the packets are received with delays more than the absolute delay requirement (which is 30 ms in Figure 5.12). According to the authors in [70], delay compensation methods that are implemented by the two communicating parties can accommodate for these few cases and eventually a stable haptic interaction is guaranteed.

Table 5.2 Comparing delays/jitters for parallel and multiplexed channels (delay 30 ms and jitter 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	103.95	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

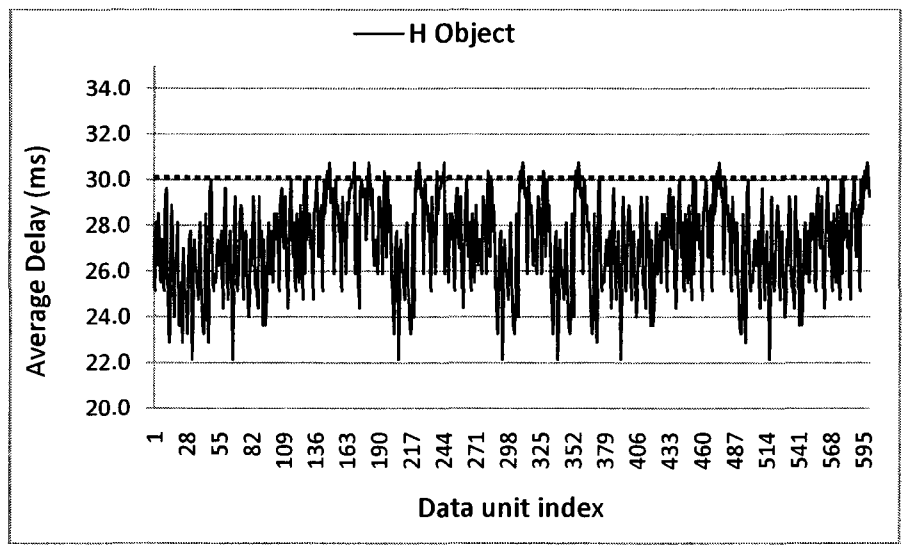

Figure 5.12: Delay variations for the haptic stream.

5.4.3.2 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. Figure 5.13 shows how the weighting factors (w_j) are converging to the desired values (x_j) 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 resources are redistributed based on the priority level of the respective media and network conditions.

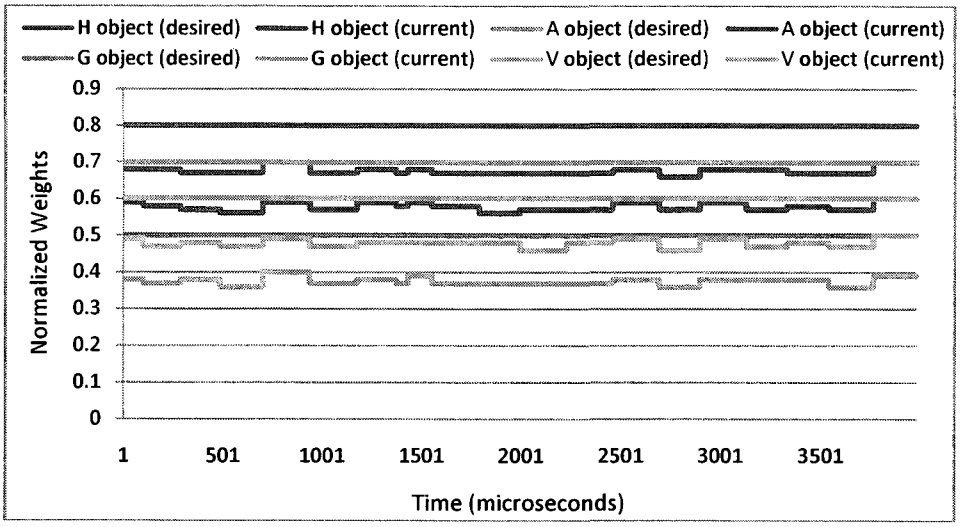

Figure 5.13 Simulation of HugMe with 4 channels (adaptability of the multiplexer).

5.4.3.3 Time Complexity Analysis

One of the critical factors to examine is the real implementation of Admux is the time complexity of the multiplexing scheme. We implemented in the `NetworkManager` class a high precision timer to measure the computation time for a multiplexing cycle. As the results in Figure 5.14 demonstrate, the computation time is

converging to about 1.87 μs which is comfortably below the 1 ms delay needed for the haptic modality. This implies that the time overhead caused by the multiplexing scheme is minimal and has no tangible overhead impact on the communication quality.

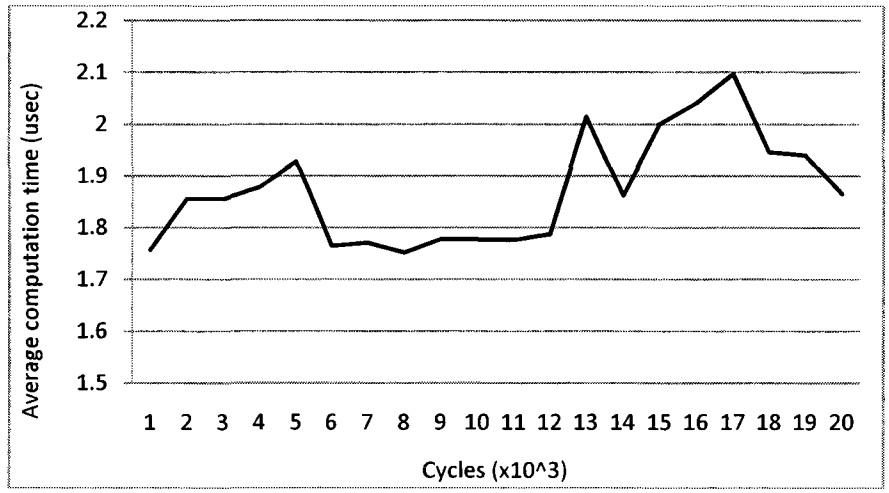

Figure 5.14: Time complexity analysis.

5.4.3.4 Application Adaptability and Multiple Buffering Scheme

The objective of this simulation is to show that Admux adapts the multiplexing scheme to the application events and needs (interactions). We simulated a collision event between the robotic arm (H object) and the virtual iris (G object). In this case, the physician should pay better attention to the visual aspects of suturing and thus the V object becomes of higher priority level than the A object. Admux should provide higher bandwidth proportion to the V object compared to the A object as long as collision event is about to happen again. We measure the dynamic bandwidth allocation based on the number of packet sent for each media channel.

The simulation results are shown in Figure 5.15 where the number of communicated packets is plotted against time in two cases: when the application event (collision event) was considered by Admux versus when it was not. In case the collision event is not considered, the A object received higher bandwidth than the V object since the audio channel has higher priority than the video channel (A object has much larger number of packet transmitted). On the other hand, when the collision event is simulated at t = 40 sec (Figure 5.15), Admux dynamically changed the prioritization of the two media channels and now the V object is allocated higher bandwidth than the A object. This demonstrates that Admux adapts the bandwidth allocation for different media channels based on the application events and interactions.

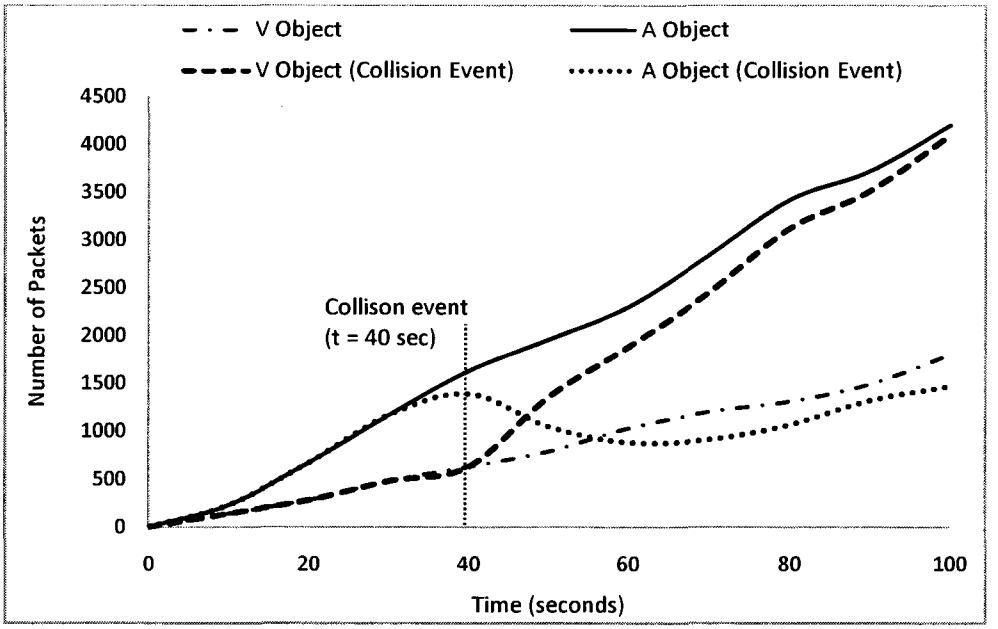

Figure 5.15: Packet throughput with collision event simulated.

5.4.3.5 Media Channels Scalability

The scalability test is performed to examine the effects 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. As shown in Figure 5.16, 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 (simulated with up to 100 channels, as shown in Figure 5.16).

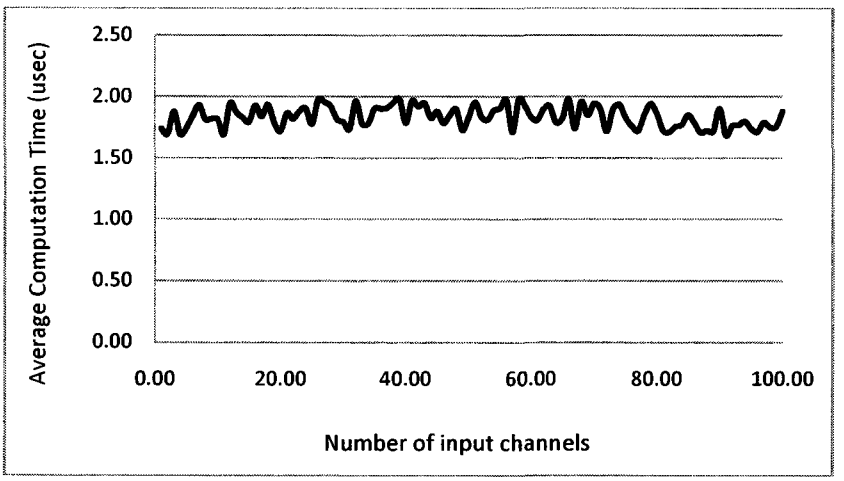

Figure 5.16: Scalability of input media channels.

5.4.3.6 TS Packet Size

In this simulation, we study the effects of the TS packet size on the average delays and jitter for the media channels. The objective here is to find the optimal size of the TS packet that would minimize the network delays and jitters. The results are shown in Figure 5.17. As we can see, the network delay and jitter are decreasing as the TS packet size increase since there is less fragmentation/defragmentation delays with larger

packet size. We behavior is true until the packet size becomes around 10000 bytes, after which the delay starts increasing again (shown in Figure 5.17).

All the experimental simulations presented in this chapter demonstrate that Admux meets all the five requirements defined in chapter 1. Therefore, the proposed communication framework of Admux has met is design objectives.

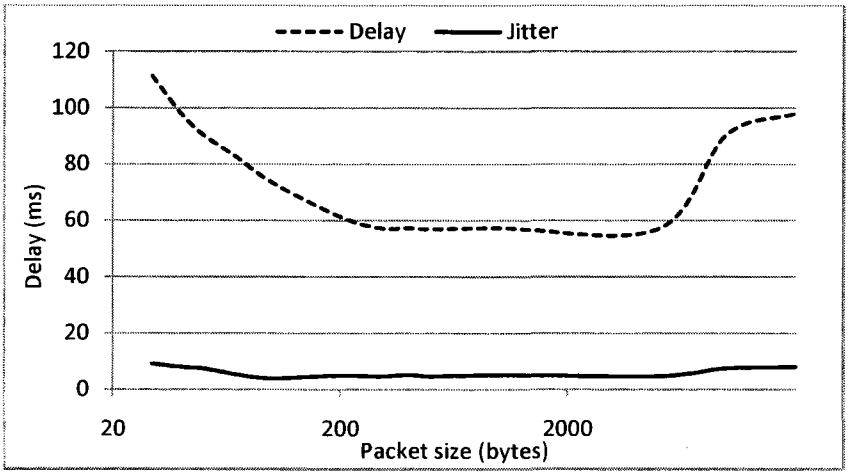

Figure 5.17: Delay/jitter variations with TS packet size.

5.5 Summary

Chapter 5 introduced the mathematical model for the multiplexing scheme that is adaptive to both the network conditions and application interactions and events. Furthermore, we performed various experimental simulations to prove the effectiveness and the capabilities of the model. Finally, we simulated the HugMe application media channels and confirmed the performance we got with the simulation results. The next chapter presents implementation details of the HugMe application and the performance evaluation of Admux when tested with HugMe and the Internet network.

CHAPTER 6

Performance Evaluation:

HugMe Interpersonal Communication System

In this chapter, we demonstrate the effectiveness of Admux as a communication framework for haptic-enabled interpersonal communication system. The test application is named HugMe and involves the communication of haptic, audio, video, and graphic media. Notice that such application is not critical and thus does not require strict network conditions and therefore the use of the Internet as the communication medium is justified due to its high availability and low cost.

6.1 HugMe: Haptic Interpersonal Communication

The incorporation of force feedback in synchronous teleconferencing multimedia systems has been challenged by the high haptic servo loop (typically 1 kHz), consistency assurance, access control, transparency, and stability, among others. Existing research about interpersonal touch over a network can be categorized as synchronous and asynchronous. One of the early asynchronous systems was the Taptap prototype [71], which is a wearable haptic system that allows nurturing human touch to be recorded, broadcast and played back for emotional therapy. The tactile data is transmitted asynchronously.

Another commercially available hug shirt that enables people feel hugs over distance is proposed in [72]. The shirt embeds sensors/actuators to read/recreate the sensation of touch, skin warmth, and emotion of the hug (heartbeat rate), sent by a distant

lover. The hugger sends hugs using a Java-enabled mobile phone application, in an as easy way as an SMS is sent, through the mobile network to the loved one's mobile phone, which in turn delivers the hug message to the shirt via Bluetooth.

As for synchronous interaction paradigms, a tele-haptic body touching system that enables parent/child interaction is described in [73]. An Internet pajama is envisioned to promote physical closeness between remote parent and child. The pajama reproduces hugging sensation that the parent applies to a doll or teddy bear, to the child. A similar idea is presented in [74] where a human/poultry and poultry/human interaction is made possible using a doll, which resembles the remote pet, and a tactile suit that is put on the pet body.

Unlike most of the previous works, HugMe system enables touch interaction that is synchronized with audio/visual information. It is worth mentioning here that the HugMe system is not meant to replace human-human contact, but to enhance the physical intimacy in the remote interaction between lovers whenever they cannot physically meet for some reason. It has other interesting applications in the medical field especially with children and elderly [75].

Figure 6.1 shows the GUI of the implemented HugMe system. The HugMe system enables touch interaction that is synchronized with audio/visual information. In this application, there will be four major data streams (haptic, video, depth, and graphics) where the transmission of media data is handled by Admux. The objective of using this application is to evaluate the performance of Admux in terms of customizability and flexibility to satisfy the varying and sometimes conflicting requirements of haptic-audio-visual applications.

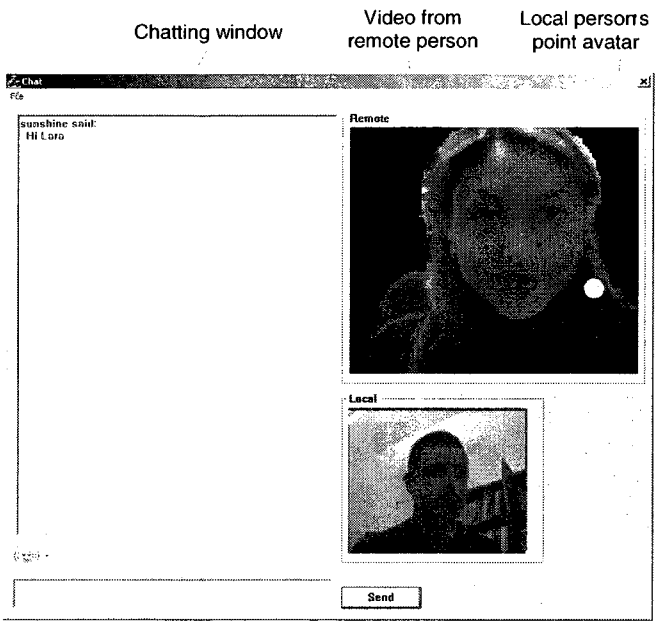

Figure 6.1: HugMe system application.

Figure 6.2 shows the implemented HugMe system with local and remote users. At the remote site, the remote person is captured with the depth camera, Zcam^{TM}, and the marker positions corresponding to the chest and the upper arm are computed through the augmented reality toolkit (ARToolKit) and these data are transmitted to the local side. The local person can 3-dimensionally see and touch the remote person through a 3 degree-of-freedom force feedback device, the Novint Falcon [58]. When the sphere avatar, that represents the position of the human hand in the scene, collides with the 2.5D remote person, the local person can feel the contact force and the contact position on the human model of the remote person is computed and transmitted to the remote side. This contact position is used to stimulate the remote person on the jacket embedded with the vibrating motors.

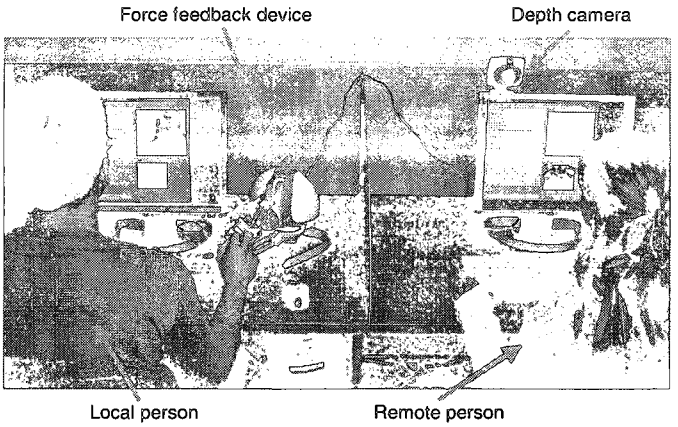

Figure 6.2: HugMe system with local and remote users.

6.2 HugMe System Implementation

This section describes a one-way version of the HugMe system where the parent is trying to touch the body of his/her child. The same system can be duplicated to enable the mutual touching between the two users. Figure 6.3 shows the system block diagram at the child's side whereas Figure 6.4 shows the system diagram at the parent's side. In the following, we briefly describe the comprising components of the HugMe software architecture and its implementation.

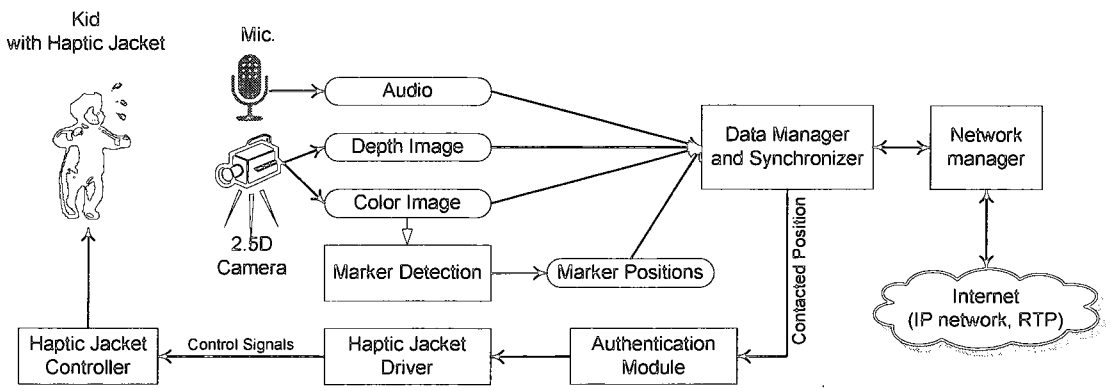

Figure 6.3: System block diagram on the child side.

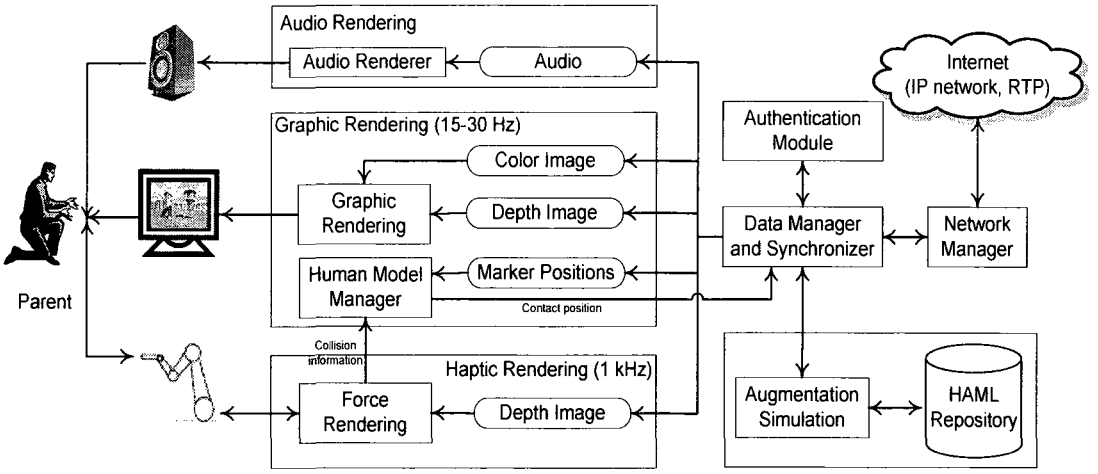

Figure 6.4: System on the parent side.

6.2.1 Depth Video Camera

The depth video camera is capable of generating RGB and D (Depth) signals. The depth signal is a grey-scale bitmap image where each pixel value represents the distance between the camera and the corresponding pixel in the RGB image. The concept of operation is simple: A light beam is generated using a square laser pulse and transmitted along the Field Of View (FOV) of the camera. The reflected beam carries an imprint of the objects depth information. The depth information is extracted using a fast image shutter. A commercially available camera that serves this concept is the Z-Cam^{TM}, developed and marketed by 3DV Systems [76].

6.2.2 Graphic and Haptic Rendering

The graphic rendering module renders the 3D scene using OpenGL. All the pixels of the depth image are transformed into 3D space by using camera parameters and triangulated (with low resolution for fast rendering) and the color image is mapped on it

as a texture. Since the captured scene is transformed into 3D space, we can produce the stereoscopic view. The haptic rendering module calculates 3D interaction force between the transformed depth image and the device position [77]. As a result, parent can touch the video capturing the child.

6.2.3 Marker Detector

In order to map the collision point in the haptic rendering and the touched point in real child, we need to track touchable part of the child. This component is responsible for tracking the movement of the remote user which can be used to construct a real-time representation (avatar). For instance, one possible tool that can be used is the Augmented Reality Tool Kit (ARToolKit) [78] that optically tracks markers, attached on touchable part of the child, in the images, and this information is mapped into the 3D depth image space. By doing this, we can transform the collision information in the haptic rendering algorithm into the touched point on the child's body, namely the actuation point of the haptic jacket.

6.2.4 Human Model Manager

The human model manager keeps track of the user body position and calculates the touched point on the human model. This is accomplished by continuously sending the updated positions of the markers (at a rate of 30-60 Hz). The human model manager maintains a graphical representation of the remote user using a set of graphic primitives. The positions and/or orientations of these primitives are updated every time the remote user moves. The human model manager is consulted by the haptic rendering component to check for possible collision between the haptic device and the user model. Therefore,

the haptic rendering is performed locally given the updated representation of the remote user. In this implementation, the torso is modeled with a rectangular parallelepiped and the right upper and lower arms with cylinders for simple calculation as a proof-of-concept.

6.2.5 Haptic Jacket

The haptic jacket is a suit that embeds vibro-tactile actuators to simulate the sense of touch. One possible design is to use a network of tiny vibrating motors distributed over a flexible layer. In order to simulate the feeling of touch, the different actuators should be controlled in a manner that best matches the real touch or touch stroke to be initiated. For example, to simulate a poke, a concentric layout of motors may be used, where a center actuator simulates the poke touch while other circularly distributed motors form the surrounding concentric circles. In addition, we plan to use heaters to simulate the warmth of touch.

Fig. 4 shows the haptic jacket that is embedded with arrays of vibrating motors. In order to measure the positions of the chest part and the upper arm, two different markers were attached on the middle of the chest and the upper arm as shown in Figure 5(a). For easier maintenance, the arm part of the jacket was cut along and the zipper is attached along the cut line, and then the array of vibrating motors was attached on the inner part of the jacket and one layer of inner fabric arm was attached to prevent the vibrating motors and the electric lines directly touch the skin. Same approach is applied to the chest part: the array of vibrating motors was attached on the inner part and a layer of inner fabric patch was attached to zip them together. Figure 5(b) and (c) show the embedded vibrating motors zipped open. Yellow lines show the zipper lines.

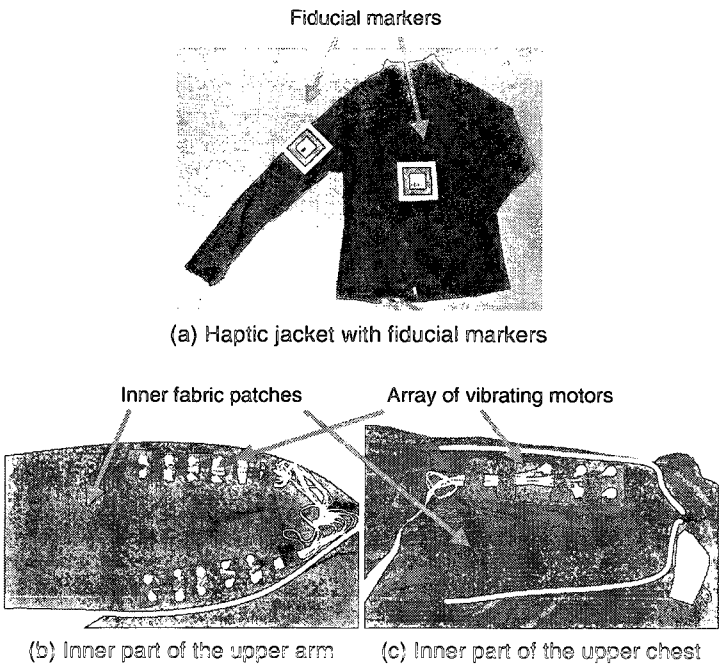

Figure 6.5: Haptic jacket.

6.2.6 Haptic Interface

The haptic interface behaves like small robots that exchange mechanical energy with users (reads handler position and sends back forces). It provides the parent with the touch feeling whenever the haptic device end effecter collides with any object of the remote environment (in our case the video contents). The HugMe system used the Falcon device, developed and marketed by Novint technologies, Inc. It was chosen for this application since it can be afforded for personal use – it costs less than $200 US.

6.2.7 Network Manager

The network manager takes care of transmitting and receiving the multimedia data from one end to the other. Here is where Admux is deployed; on top the UDP

transport protocol. This component is responsible for the delivery of the color information, the depth information, marker positions, and contact information (haptic data).

6.3 HugMe Software Implementation

The HugMe software system is divided into three subsystems: the HugMe subsystem, the GUI subsystem, and the Admux subsystem. The UML package diagram for the overall software implementation is shown in Figure 6.6. The HugMe subsystem uses both the GUI package for rendering the multimedia contents to the user, and the Admux package for communicating the contents with the remote user.

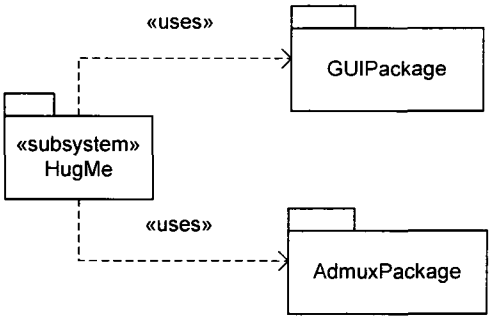

Figure 6.6: HugMe UML package diagram.

6.3.1 HugMe Subsystem Implementation

The HugMe subsystem provides various interfaces with the input/output devices such as the haptic device, the depth camera, the haptic jacket, etc. Moreover, the HugMe subsystem maintains the graphics and haptic rendering and compression (we used H263 compression for video and depth streams), in addition to the human model that is used for human body tracking. The HugMe class is the core component of this subsystem; it

communicates the system state with the communication subsystem (the Admux subsystem) and the presentation subsystem (GUI subsystem).

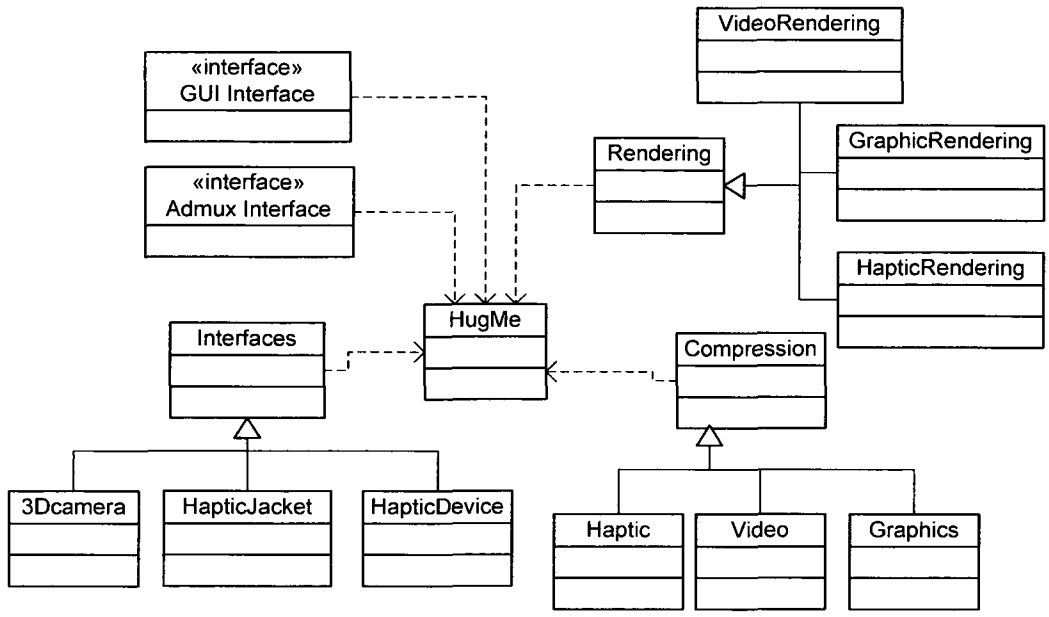

Figure 6.7: HugMe subsystem UML class diagram.

6.3.2 GUI Subsystem Implementation

The GUI subsystem encapsulates the GUI components that display various multimedia information including haptic, video, text, and graphics to the user. The primary interface component is the ChatDlg class that renders the main window of the HugMe system. It renders the video and depth video streams, the haptic device end effecter point, and the chatting components. Moreover, the GUI subsystem hosts other GUI components for setting up the network connection (ConnectDlg class), the configurations for input/output devices (OptionsDlg class), and the displaying emotional characters (EmotionsDlg class).

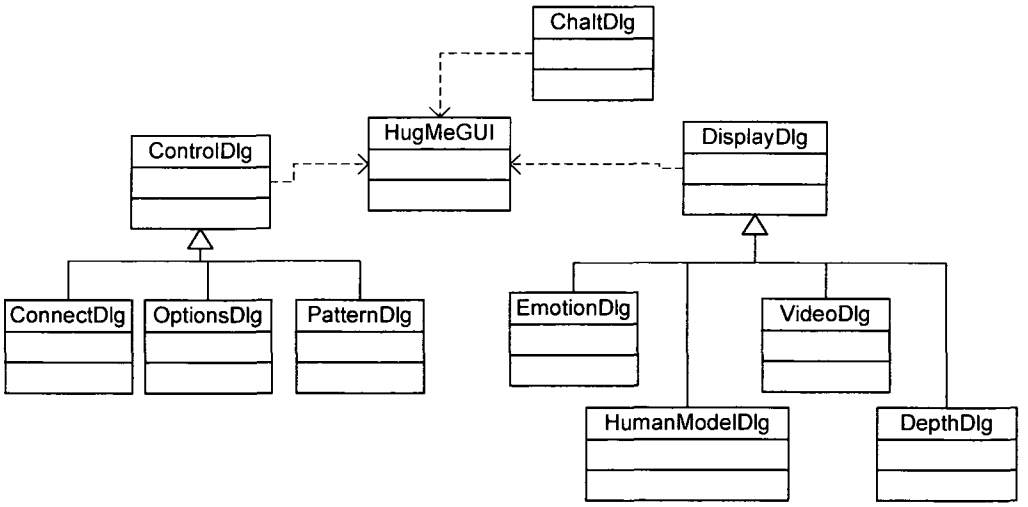

Figure 6.8: UML class diagram for the GUI subsystem.

6.4 Experimental Test bed and Metrics

The experimental test bed is composed of two Pentium 4 PCs with 2 Gb RAM and 100Mbs Ethernet cards, a haptic device, a 3D camera, and a haptic jacket. The haptic device we used in this experimentation is the Falcon device, developed and marketed by Novint technologies, Inc. The haptic jacket is developed at the MCRlab and published in our previous work [77]. Finally, the camera used for generating color and depth information is the Z-Cam™, developed and marketed by 3DV Systems. The experiment was conducted over the Internet network between two hosts in the same city (Ottawa, Canada). The average delay between the two hosts was computed to be 40ms and the jitter was 6ms during the time of performing the experiment.

In order to measure precisely the network conditions, the clocks of both workstations were synchronized. For this purpose, a Network Time Protocol (NTP) server is used where both workstations maintain a connection with the NTP server in

order to synchronize their clocks. The clock synchronization precision is comfortably maintained within one millisecond (particularly important for haptic media synchronization).

The four channels that we incorporated in the experiment were the haptic channel (sending the haptic interface position information), the video channel (sending the RGB frames), the depth channel (sending the depth frames), and a graphics channel (sending the human model information, the haptic jacket commands, and the chatting text). The delay requirements for the four channels are as follows; haptic: 20 ms, video: 33 ms, depth: 33ms, and graphics: 400ms.

6.5 Performance Results

6.5.1 Multiplexing Scheme versus Parallel Communication

The objective here is to test the performance of Admux with the Internet network and eventually confirm the simulation results. Four media channels were used in this study haptic object (H object), video object (V object), depth object (D object), and graphics object (G object). Both the V object and the D object have same characteristics, and thus should be treated equally by Admux. The H object is assigned the highest priority (that might change at run-time depending on the application interactions). Finally, the G object has the least priority since chatting/graphics data has the most relaxed QoS requirements among all the other channels. Similarly with what we did in the simulation, we compare the delay and jitter values for each channel when parallel UDP communication is used versus when the multiplexer is used. In this case, we measured the average network delay and jitter of the Internet network and found them 26ms and 6ms respectively.

As shown in Table 6.1, it is clear that the average delays and jitters are almost equal in the case of using parallel UDP channels. This is because UDP is best effort and does not guarantee any prioritization for the QoS resources allocation. On the other hand, Admux enables the application to assign network resources in proportion to the media needs and application requirements. In this case, all the delays of the four media are met whereas with parallel channels, the delay and jitter requirements were not met for the haptic media. This shows that Admux can adapt to the application needs, particularly when the QoS requirements for each media channel is very different from other media channels.

Table 6.1: Comparing delays/jitters for parallel and multiplexed channels (delay 26 ms and jitter 6 ms).

	Approach	**Haptic**	**Graphics**	**Video**	**Depth**
Average Frame size (bytes)		32	200	14K	14K
Frame rate (fps)		200	33	30	30
Delay (ms)	Multiplexer	28.71	51.54	268.54	266.85
	Parallel	63.58	69.88	72.14	71.9
	Desirable	40	100	300	300
Jitter (ms)	Multiplexer	7.98	18.98	15.62	14.53
	Parallel	11.15	11.12	12.79	12.28
	Desirable	10	50	30	30

Another interesting case is when the network cannot provide the minimum requirements for a particular channel. We have shown in Chapter 5 that Admux will drop that channel and allocate all its resources to other channels (so that their minimum requirements are better met). We changed the desired threshold for the D object so that

the network will fail to meet the minimum requirement and checked if any data from the D object channel will be transmitted. As expected Admux has dropped the D object channel and now the other three channels can comfortably meet their requirements.

6.5.2 Time Complexity Analysis

One of the critical factors to confirm with the HugMe system is the time complexity of the multiplexing scheme. We used the high precision timer, implemented in the `NetworkManager` class, to measure the computation time for a complete multiplexing cycle. Figure 6.9 demonstrates that the computation time is converging to 2.005 μs which is comfortably below the 1 ms delay needed for the haptic modality. Comparing this result with the simulation we got in chapter 5 (average computation time was found to be 1.87 μs, we noticed a marginal difference that is probably due to copying the media data into temporary buffers (to ensure synchronization). In all means, the time overhead caused by the multiplexing scheme is comfortably negligible and eventually has no tangible overhead impact on the communication quality.

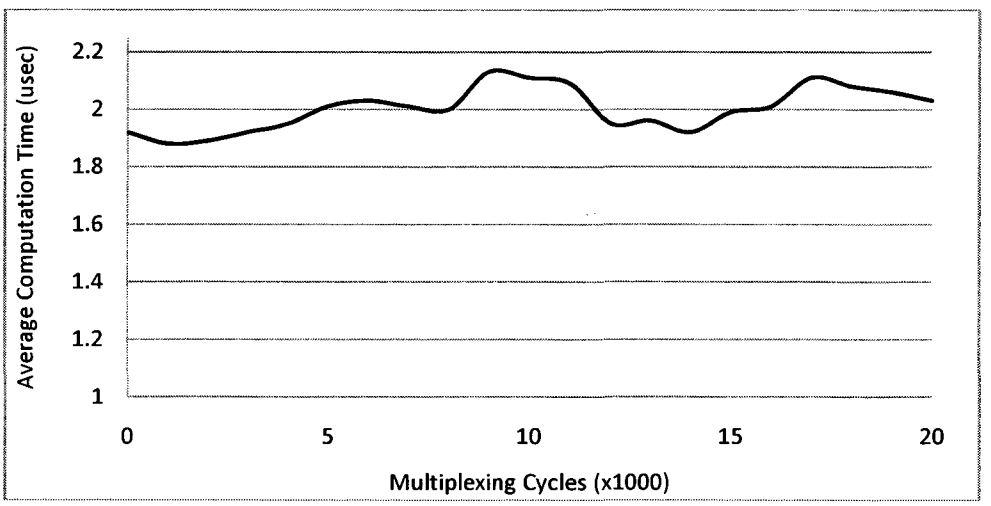

Figure 6.9: Time complexity analysis with HugMe system.

6.5.3 TS Packet Size and Error Rate

One observation that we noticed during the implementation was the interdependence between the TS packet size and the error rate for each media channel. Therefore, we conducted a study to examine that relationship by measuring the per channel error rate as function of the TS packet size. The results are shown in Figure 6.10. The figure shows that the error rate for the V object and the D object data are decreasing as the size of the TS packet increases. This is because the packetization of these media data is resulting in less number of fragments per frame, and thus a complete frame is interleaved along less number of UDP packets. On the other hand, the H object showed a different relationship; the average error rate was not affected by the change in the TS packet size. The reason is because the haptic frame is very small (only 32 bytes) and there was no fragmentation for this media. It is worth mentioning that even though smaller size TS packet has resulted in smaller values for the average error rate, but the V object and D object have suffered larger delays.

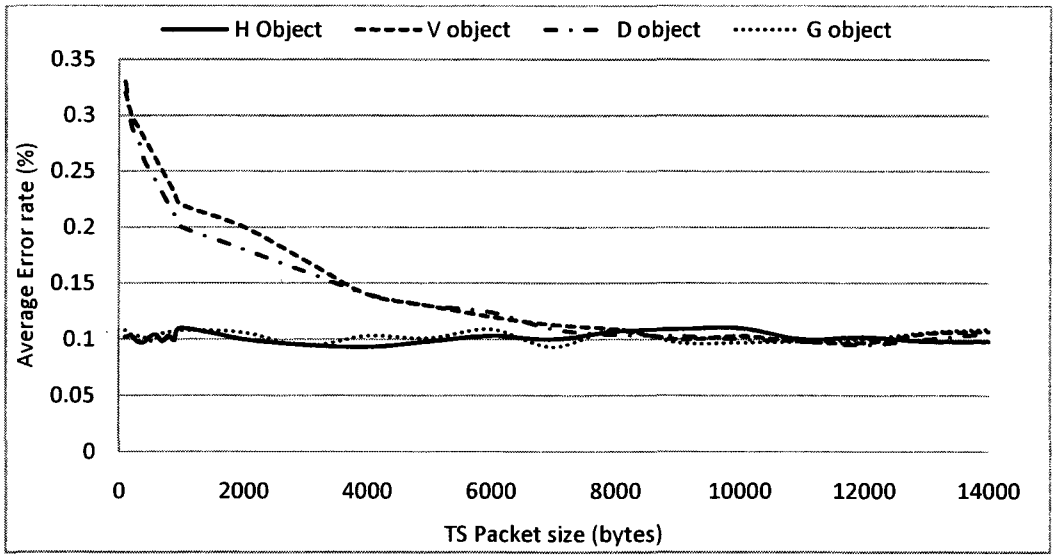

Figure 6.10: Average error rate (%) versus the TS packet size.

6.5.4 TS Packet Size Optimization

In this study, we examine the effects of the TS packet size on the average delays and jitter for the four media channels. The objective here is to find the optimal size of the TS packet in order to tradeoff the network delays and jitters and the average error rate (as shown previously in section 6.4.3). The results are shown in Figure 6.11. The figure shows that the network delay and jitter are decreasing as the TS packet size increase since there is less fragmentation/defragmentation delays with larger packet size. This behavior is true until the packet size becomes around 5 Kbytes, after which the delay starts increasing again (shown in Figure 6.11).

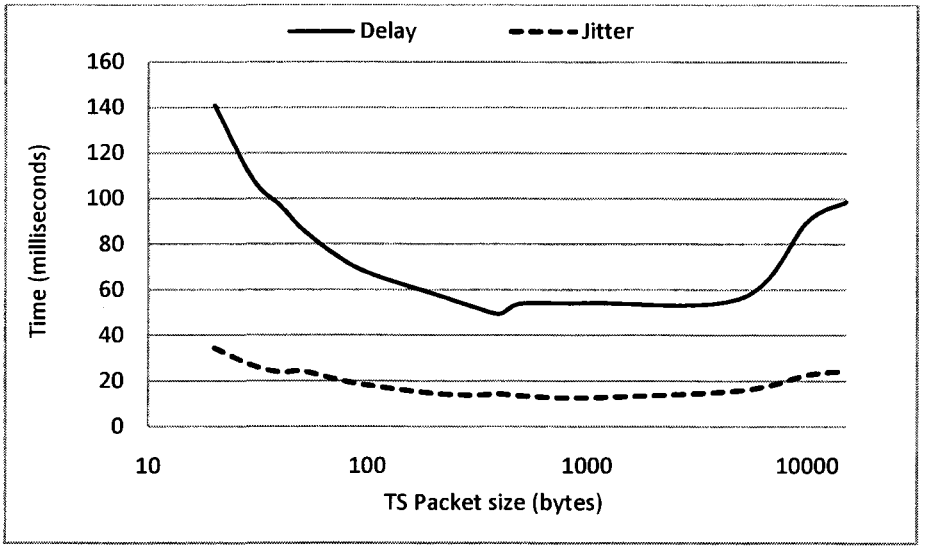

Figure 6.11 Delay/jitter variations with TS packet size.

6.6 Discussion

Our performance evaluation with the HugMe system has demonstrated three distinguished capabilities of Admux. First, Admux provides a mechanism for the synchronization of haptic data with audio/video data. Second, Admux is highly reconfigurable and loosely coupled with the application as it uses HAML to

communicate the application requirements and the multiplexing settings. That is why Admux is placed at the application layer to provide the application with higher control over the communication infrastructure. Third, Admux adapts to application level events and interactions as well as the network conditions. Adapting to the network conditions is of significant importance for non-dedicated networks such as the Internet. This is why Admux is originally designed for the Internet network, since dedicated networks have fixed resources and thus they do not require any adaptive communication protocols. With all the apparent merits of Admux, there are few comments that should be made here:

- The proposed multiplexing scheme is statistical. Therefore no absolute QoS values are guaranteed! This has a significant impact for haptic media since the stability of the haptic rendering might be affected. Therefore, Admux should be used in conjunction with delay compensation and jitter smoothing algorithms to guarantee that stability of haptic interaction. Even though this requirement is important for the application in general, we do not consider it part of Admux design and thus we used existing work to compensate for excessive haptic delays. Some applications have strict requirements and thus cannot use Admux; in such cases these applications usually use dedicated networks.
- Admux assumes that the application owns and manages HAML document describing its requirements and specifications – and in particular the communication requirements. This implies that applications that do not support HAML cannot use Admux. It is worth mentioning here that HAML provides way more capabilities for the haptic-audio-visual application and thus it is a safe assumption to make that several application would utilize it.

- One interesting finding that we noticed when performing the performance evaluation with HugMe system was the error detection and correction of for PES packets. Since Admux uses UDP and thus encounters unreliable communication, there is a need for error resilience algorithm that can reassemble PES packets even with the case of loosing TS packets. The current implementation ignores this problem and drops the whole PES packet if at least one TS packet was lost. As part of the future work, an error resilience algorithm should be integrated into Admux to decrease the drop rate of PES frames and thus improve the network utilization the overall quality of experience.

As a final remark, all the experimental results presented in this chapter demonstrate that Admux meets all the five requirements defined in chapter 1. Therefore, the proposed communication framework of Admux has met our design objectives.

CHAPTER 7

Conclusions and Future Works

In this research, we have presented the design, implementation, and evaluation of a communication framework for haptic-audio-visual multimedia systems, named Admux. The design, mathematical modeling, and implementation of an adaptive multiplexing scheme are also proposed. Furthermore, an interpersonal communication system, named HugMe system, has been both simulated and implemented 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 and events.

7.1 Thesis Summary

Admux is a powerful communication framework for synchronous haptic-audio-video applications that is adaptive to both the application requirements and network conditions. These requirements can abstractly be communicated with Admux as it supports HAML description; and thus Admux is loosely coupled from the application design. Furthermore, extensive simulations are performed, including time complexity, scalability, and adaptability to application requirements and network conditions, to analyze and verify the behavior of the mathematical model that we developed for Admux.

The implementation of the application scenario (the HugMe system) has shown that the size of the TS packet has significant impact on the performance. Furthermore, error resilience algorithms should be integrated into Admux to enhance its performance

against network packets loss. As shown by both the simulation and experimental testing, Admux was able to distribute network resources in proportion to the QoS requirements for each media type.

7.2 Future Work

Although significant progress has been made in this thesis, a multitude of challenges remains to be addressed. Apart from further testing and performance evaluation, we list a few future directions here:

- *Multi-User Interaction:* One possible research direction is to investigate the potential for multi-user communication with Admux. In this case, the communication is dependent on the application interactions (that are now collaborative) and thus the interactions at each communicating party should be taken into consideration. Furthermore, the prioritization for different media channels should be assigned to optimize the quality of collaboration. Multi-user interaction involves various communication paradigms such as client-server and peer-to-peer architectures. This is of significant impact on applications such as gaming and entertainment where multiple users are expected to participate in the same game environment and thus scalability becomes a fundamental research question.

- *Error Resilience:* The current implementation of Admux does not include any mechanism for error resilience. Therefore, future research is needed with issues such as accurate error models and their effect on system performance, in case of multimedia transmission over the internet – and possibly wireless networks.

- *Adaptive Compression:* The ability of a communication framework to control the compression settings based on the network and application requirements enhances the

optimization of the communication. Even though this factor was incorporated in Admux framework, it was not part of the implementation effort. Therefore, implementing this facility and investigating a control algorithm that would adapt the compression for different media according to the communication requirements would have a significant impact on the quality of communication.

- *Standardizing HAML:* We are currently in the process of incorporating HAML in standardization bodies such as MPEG-V and H3D. This effort will go on as one future direction for HAML in order to standardize the descriptions of haptic-audio-video applications.

- *Testing Admux with diverse applications*: Applications from different fields (other than chatting) will be considered for Admux performance. Such studies will experimentally prove the ability of Admux to adapt to various haptic applications with different networking requirements and interaction styles.

Bibliography

[1.]R. MacLaverty and I. Defee, "Multimodal interaction in multimedia applications", In proceedings of the IEEE First Workshop on Multimedia Signal Processing, page(s): 25-30, Princeton, USA, 1994.

[2.]P. Milgram and F. Kishino, "A Taxonomy of Mixed Reality Visual Displays", IEICE Transactions on Information Systems, Vol E77-D, No.12, 1994.

[3.]S. Andrei, 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 Proceedings of the IEEE Visualization conference, pages: 364-368, Los Alamitos, California, USA, 1994.

[4.][National Association of Broadcasters. Princeton Electronic Billboard Develops "Virtual Arena" Technology for Television. Washington, DC, National Association of Broadcasters, 1994.

[5.]B. Chebbi, D. Lazaroff, F. Bogsany, P. X. Liu, N. Liya, and M. Rossi, "Design and implementation of a collaborative virtual haptic surgical training system", In proceedings of the IEEE International Conference on Mechatronics and Automation, Vol. 1, Page(s): 315 – 320, Ontario, Canada, 2005.

[6.]M. Eid, M. Orozco, and A. El Saddik, "A Guided Tour in Haptic Audio Visual Environment and Applications", International Journal of Advanced Media and Communication, vol.1, n.3, 265 – 297, 2007.

[7.]TriSenx official Website: http://www.trisenx.com/scent_dome.html, accessed March 14, 2010.

[8.]R.S. Allison, J.E. Zacher, D. Wand, and J. Shu, "Effects of network delay on a collaborative motor task with telehaptic and televisual feedback", in Proceedings of the ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI '04), pages: 375–381, Singapore, 2004.

[9.]S. Oviatt, "Ten myths of multimodal interaction", Communications of the ACM, Vol. 42, pages: 75-81, 1999.

[10.]F. Pereira and T. Ebrahimi, "The MPEG-4 Book", IMSC Press multimedia series, Prentice Hall PTR, New Jersey, NJ, 2002.

[11.] D. Walsh, C. Gunn, M. Adcock, and M. Hotchins, "Haptics Nodes for MPEG-4 BIFS (Object Surface)", ISO/AEC JTC 1/SC 29/IWG 11, MPEG2001im7409, 2001.

[12.] J. Cha, Y. Seo, Y. Kim, and J. Ryu, "An Authoring/Editing Framework for Haptic Broadcasting: Passive Haptic Interactions using MPEG-4 BIFS," In proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WorldHaptics), pages: 274-279, Tsukuba, Japan, 2007.

[13.] J. Zhou, X. Shen, and N.D. Georganas, "Haptic tele-surgery simulation", In proceedings of the IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, page(s): 99- 104, 2004.

[14.] M. Fayad and D.C. Schmidt, "Object-Oriented Application Frameworks", Communications of the ACM, Special Issue on Object-Oriented Application Frameworks, Vol. 40, No. 10, October 1997.

[15.] Reachin Display official website, www.reachin.se, accessed on March 14, 2010.

[16.] H3D official website, www.h3dapi.org, accessed on March 14, 2010.

[17.] D. Miras, "A Survey of Network QoS Needs of Advanced Internet Applications", Working Document, Internet2 QoS Working Group, 2002.

[18.] A. Marshall, K.M. Yap, and W. Yu, "Providing QoS for Networked Peers in Distributed Haptic Virtual Environments", Advances in Multimedia, Vol. 2008, 2008.

[19.] S. R. Ellis, "What are virtual environments?", IEEE Computer Graphics and Applications, Vol. 14 (1), pages: 17-22, 1994.

[20.] T.E. Whalen, S. Noel, and J. Stewart, "Measuring the Human Side of Virtual Reality", In proceedings of the International Symposium on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Lugano, Switzerland, pages: 27-29, 2003.

[21.] D. Gracanin, Y. Zhou, and L.A. DaSilva, "Quality of Service for Networked Virtual Environments", IEEE Communications Magazine, 2004.

[22.] 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 Interactions, Vol. 7, no. 4, pages: 443-460, 2000.

[23.] 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 Vol. 13, Issue 3 - Special Issue: Collaborative Virtual Environments, 2004.

[24.] R. Iglesias, E. Prada, A. Uribe, A. Garcia-Alonso, S. Casado, T. Gutierez, "Assembly simulation on collaborative haptic virtual environments", In proceedings of the 15-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen - Bory, Czech Republic, 2007.

[25.] K.S. Park and R.V. Kenyon, "Effects of network characteristics on human performance in a collaborative virtual environment Virtual Reality", IEEE International Conference on Virtual Reality (VR'99), Houston, Texas, pages: 104-111, 1999.

[26.] P. Hinterseer, E. Steinbach, and S. Chaudhuri, "Perception-Based Compression Of Haptic Data Streams Using Kalman Filters", In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pages: 23-24, 2006.

[27.] N. Sakr, J. Zhou, N.D. Georganas, J. Zhao, X. Shen, "Prediction-based Haptic Data Reduction and Compression in Tele-Mentoring Systems", In proceedings of the IEEE International Instrumentation and Measurement Technology Conference, pages: 1828-1832, Victoria, BC, Canada, 2008.

[28.] S. Lee and J. Kim, "Transparency analysis and delay compensation scheme for haptic-based networked virtual environments", Elsevier Computer Communications, Vol. 32, Issue 5, pages: 992-999, 2009.

[29.] O. Wongwirat, S. Ohara, and N. Chotikakamthorn, "An Adaptive Buffer Control using Moving Average Smoothing Technique for Haptic Media Synchronization", In proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, Issue, 12-12, pages:1334 –1340, 2005.

[30.] P. Hinterseer and E. Steinbach, "A Psychophysically Motivated Compression Approach for 3D Haptic Data", In Proceedings of the Symposium on Haptic interfaces For Virtual Environment and Teleoperator Systems, Washington, DC, 2006.

[31.] C. Shahabi, A. Ortega, and M.R. Kolahdouzan, 'A comparison of different haptic compression techniques', In proceedings of the IEEE International Conference on Multimedia and Expo, Vol. 1, Pages: 657 – 660, 2002.

[32.] N. Sakr, J. Zhou, N.D. Georganas, J. Zhao, E.M. Petriu, "Robust Perception-based Data Reduction and Transmission in Telehaptic Systems", In proceedings of the World Haptics, Third Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Salt Lake City, USA, 2009.

[33.] R. J. Anderson and M. W. Spong, "Bilateral control of teleoperators with time delay", IEEE Transactions on Automatic Control, Vol. 34(5), pages: 494–501, 1989.

[34.] H. Cho and J. Park, "Impedance control with variable damping for bilateral teleoperation under time delay", JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing, Vol. 48(4), pages: 695–703, 2005.

[35.] K. Hashtrudi-Zaad and S. Salcudean, "Analysis of control architectures for teleoperation systems with impedance/admittance master and slave manipulators", International Journal of Robotics Research, Vol. 20(6), pages: 419–445, 2001.

[36.] G. Niemeyer and J.J. Slotine, "Stable adaptive teleoperation", IEEE Journal of Oceanic Engineering, Vol. 16(1), pages: 152–162, 1991.

[37.] P. Hinterseer, E. Steinbach, S. Hirche, and M. Buss, "A novel psychophysically motivated transmission approach for haptic data streams in telepresence and teleaction systems," In proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, , Vol. 2, pages: 1097–1100, 2005.

[38.] 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," IEEE Transactions on Signal Processing, vol. 56, no. 2, pp. 588–597, Feb. 2008.

[39.] R. Chaudhari, J. Kammerl, and E. Steinbach, "On the Compression and Rendering of Event-triggered Force Transients in Networked Virtual Environments", In proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and Games, Lecco, Italy, 2009.

[40.] J.D. Hwang, M.D. Williams, and G. Niemeyer, "Toward event-based haptics: Rendering contact using open-loop force pulses," In proceedings of the International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Vol. 0, pages: 24–31, 2004.

[41.] M. Mauve, V. Hilt, C. Kuhmunch, and W. Effelsberg, "RTP/I - Toward a Common Application-Level Protocol for Distributed Interactive Media". IEEE Transactions on Multimedia, Vol. 3(1), pages: 152–161, 2001.

[42.] R. Hubbold, "Collaborative stretcher carrying: a case study", In Proceedings of the Eighth Eurographics Workshop on Virtual Environments, Edinburgh, UK, 2002.

[43.] L. Chen, G. Chen, H. Chen, J. March, S. Benford, S., and Z.G. Pan, "An HCI method to improve the human performance reduced by local-lag mechanism", Interacting with Computers, Vol. 19 (2), pages: 215-224, 2007.

[44.] L. Gautier, C. Diot, and J. Kurose, "End-to-end transmission control mechanisms for multiparty interactive applications on the internet", In proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFO- COM), Vol. 3, pages: 1470-1479, 1999.

[45.] S. Dodeller, "Transport Layer Protocols for Haptic Virtual Environments", M.S. thesis, University of Ottawa, Ottawa, ON, Canada, 2004.

[46.] Z. Cen, M.W. Mutka, D. Zhu, and N. Xi, "Supermedia Transport for Teleoperations over Overlay Networks", In proceedings of the IFIP-TC6 NETWORKING Conference, Waterloo, Ontario Canada, 2005.

[47.] O. Wongwirat, S. Ohara, and N. Chotikakamthorn, "An Adaptive Buffer Control using Moving Average Smoothing Technique for Haptic Media Synchronization", In proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, Issue, 12-12, pages:1334 – 1340, 2005.

[48.] S. Lee, S. Moon, and J. Kim, "A network-adaptive transport scheme for haptic-based collaborative virtual environments", In proceedings of the 5th ACM SIGCOMM Workshop on Network and System Support for Games, Singapore, 2006.

[49.] Z. C. Mutka, M.Y. Liu, A. Goradia, and N. Xi, "QoS management of supermedia enhanced teleoperation via overlay networks", In proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages: 1630- 1635, Edmonton, Alberta, Canada, 2005.

[50.] M. Kuschel, P. Kremer, S. Hirche, and M. Buss, "Lossy data reduction methods in haptic telepresence systems," In proceedings of the IEEE International Conference on Robotic Automation, pages: 2933–2938, Orlando, FL, 2006,.

[51.] H. Al Osman, M. Eid, R. Iglesias, and A. El Saddik, "ALPHAN: Application Layer Protocol for HAptic Networking", In proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications, Page(s):96 – 101, Ottawa, Canada, 2007.

[52.] H. Al Osman, M. Eid, and A. El Saddik, "Evaluating ALPHAN: A Communication Protocol for Haptic Interaction", In proceedings of the Symposium on Haptic interfaces for virtual environment and teleoperator systems, Page(s):361 – 366, 2008.

[53.] K. Chandra, "Statistical Multiplexing", Wiley Encyclopedia of Telecommunications, 2003.

[54.] M.A. Saleh, I.W. Habib, and T.N. Saadawi, "Simulation Analysis of a Communication Link with Statistically Multiplexed Bursty Voice Sources", IEEE Journal on Selected Areas in Communications, Vol. 11, No. 3, 1993.

[55.] M. Zajeganovic-Ivancic, I.S. Reljin, B.D. Reljin, "Video Multicoder with Neural Network Control", In proceedings of the 9th Symposium on Neural Network Applications in Electrical Engineering, NEUREL, Belgrade, Serbia, 2008.

[56.] I.S. Reljin, M. StanojeviC, B.D. Reljin, "Modified Round-Robin Scheduler for Pareto Traffic Streams", In proceedings of the International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, NiS, Yugoslavia, 2001.

[57.] C. Liu, Y. Xie, M.J. Lee, and T.N. Saadawi, " Multipoint Multimedia Teleconference System with Adaptive Synchronization", IEEE Journal on Selected Areas in Communications, Vol. 14(7), pages: 1422-1435, 1998.

[58.] Novint Technologies, Inc., http://www.novint.com, accessed March 14, 2010.

[59.] SensorML developer website (2006), Available online at http://vast.uah.edu/SensorML. accessed March 14, 2010.

[60.] Transducer Markup Language official website: http://www.transducerml.org/standards.htm, accessed March 14, 2010.

[61.] Web3D Consortium, "The Virtual Reality Modeling Language", http://www.w3.org/MarkUp/VRML, accessed March 14, 2010.

[62.] Web3D Consortium, "X3D", accessed on 05/31/2006, http://www.web3d.org, accessed March 14, 2010.

[63.] J. Zhou, X. Shen, I. Shakra, A. El Saddik, and N.D. Georganas, "XML-based Representation of Haptic Information", In proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, 2005.

[64.] J. Carter, J. Van Erp, D. Fourney, S. Fukuzumi, J. Gardner, Y. Horiuchi, G. Jansson, H. Jorgensen, R. Kadefors, T. Kobayashi, M.G. Kwok, M. Miyagi, K.V. Nesbitt, "The GOTHI Model of Tactile and Haptic Interaction", In Proceedings of the Guidelines on Tactile and Haptic Interactions (GOTHI-05), October 24-26, 2005.

[65.] J. Van, J.B.F. Erp, I. Andrew, and J. Carter, "ISO's Work on Tactile and Haptic Interaction Guidelines", In proceedings of the EuroHaptics 2006, paris, France 2006.

[66.] F. R. El-Far, M. Eid, M. Orozco, and A. El Saddik, "Haptic Application Meta-Language", DS-RT, Malaga, Spain, 2006.

[67.] M. Eid, A. Alamri, and A. El Saddik, "MPEG-7 Description of Haptic Applications Using HAML". In proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, 2006.

[68.] Y. Lee, O. Min, and H. Kim, "Performance evaluation technique of the RTSP based streaming server", In proceedings of the Fourth Annual ACIS International

Conference on Computer and Information Science, Jeju Island, South Korea, 2005.

[69.] H. Al Osman, M. Eid, R. Iglesias, and A. El Saddik, "ALPHAN: Application Layer Protocol for HAptic Networking", In proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, Page(s):96 – 101, 2007.

[70.] I. Md. Wahidul, "Stability of haptic displays in distributed virtual environment", Masters thesis, Concordia University, 2001.

[71.] L. Bonanni, C. Vaucelle, J. Lieberman, and O. Zuckerman, "TapTap: a haptic wearable for asynchronous distributed touch therapy", In proceedings of the 7th international conference on Human factors in computing systems, 2006.

[72.] CuteCircuit official website: http://www.cutecircuit.com/projects/wearables/thehugshirt/, accessed March 14, 2010.

[73.] J. Teh, S.P. Lee, and A.D. Cheok, "Internet pajama", In Proc. of International Conference on E-Learning and Games (Edutainment), pages: 1288-1291, California, 2006.

[74.] Teh, K.S., Lee, S.P., and Cheok, A.D., "Poultry Internet: a remote human-pet interaction system", In proceedings of the NetGames & 5th Workshop on Network & System Support for Games, Singapore, 2006.

[75.] Adoption Media, "The Importance of Touch", http://library.adoption.com/Parenting-Skills/The-Importanc e-of-Touch/article/3060/1.html, (accessed March 7, 2010).

[76.] 3DV Systems, http://www.3dvsystems.com, accessed March 14, 2010.

[77.] J. Cha, M. Eid, and A. El Saddik, "DIBHR: Depth Image-Based Haptic Rendering" In proceedings of the EuroHaptics conference, pages: 640-650, Madrid, Spain, 2008.

[78.] AR Toolkit website, http://www.hitl.washington.edu/artoolkit, accessed March 14, 2010.