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

Abu Hossain

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Virtual Reality Simulation Modeling for Tele-Surgery and Tele-Haptic Class of Application

TITRE DE LA THÈSE / TITLE OF THESIS

A. Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

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

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

S. Shirmohammadi

G. Wainer

Gary W. Slater

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

Virtual Reality Simulation Modeling for Tele-Surgery and Tele-Haptic Class of Applications

by

Abu Hossain, B.E.

A Master's thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master's of Applied Science in
Electrical Engineering

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-18425-7
Our file Notre référence
ISBN: 978-0-494-18425-7

Abstract

The incorporation of Haptic interfaces in Collaborative Virtual Environments (CVE) suffers from setbacks due to an inherent technical problem that occurs when the users are geographically distributed. The main causes of such discrepancies are *network delay*, a *lack of view of individual participants' activities*, and *haptic feedback*. Although some strategies exist for dealing with these concerns, they do not adequately address reality, causality, or the sense of co-presence in CVEs during closely coupled haptic tasks. In this thesis, an approach based on both decorators and prediction to compensate for network delays and lost updates is proposed. This approach can be used together with existing networking-level techniques and can improve the quality of collaboration as perceived by remote users. This work makes use of the SCTP protocol, which sends critical update messages reliably and normal update messages with best-effort. A history buffer and a decorator-based predictor are added to the receiving side.

Acknowledgements

My sincere gratitude goes to my academic supervisor, Dr. A. Boukerche, who helped, encouraged, and guided me towards my academic and personal success. Thanks for the financial support he has provided to me. The thesis would also not be successful without the help of Dr. S. Shirmohammadi. He provided me a substantial amount of help and suggestions throughout my work and allowed me to use haptic equipments that were available at the DISCOVER Laboratory. Special gratitude also goes to Richard Pazzi, Lab Manager of PARADISE Research Laboratory and Francois Malric, Lab Manager, the DISCOVER Laboratory, and fellow students of both PARADISE and DISCOVER Research Laboratories team for many helpful discussions and their constant encouragement and support.

I would like to give my sincere thanks to my wife and child for their support throughout my graduate studies. I am also grateful to all my friends and family, who helped me either with suggestions or with positive affirmation.

This work was partially supported by ORNEC, NSERC, Canada Research Chair Program, and EAR/PREA Award.

Table of Contents

List of Figures

List of Abbreviations

API: Application Programming Interface

ATOAST: Adaptive Test and Optimization of Applications in Surgery and Training

AWT: Abstract Window Toolkit

CVE: Collaborative Virtual Environment

DDM: Data Distribution Management

DISCOVER Lab: Distributed and Collaborative Virtual Environments Research Laboratory

DLL: Dynamic Loadable Library

DOF: Degree of Freedom

DVE: Distributed Virtual Environment

FIFO: First in First Out

GHOST®: General Haptic Open Software Toolkit

GPS: Global Positioning System

GUI: Graphical User Interface

HDAPI: Haptic Device API

HLA: High Level Architecture

HLAPI: Haptic Library API

HRTC: Haptic Real Time Controller

I/O: Input/Output

IP: Internet Protocol

JNI: Java Native Interface

JMF: Java Media Framework

LLC: Logical Link Control

MAC: Media Access Control

OMT: Object Model Template

OSI: Open System Interconnection

PARADISE: PARAllel, Distributed and Interactive Simulation of LargE scale Systems and Wireless and Mobile Networking Research Laboratory

PDD: PHANTOM Device Drivers,

P2P: Peer-to-Peer

RTI: Runtime Infrastructure

SCM: Session Control Manager

SCTP: Synchronous Collaboration Transport Protocol.

SRM: Scalable Reliable Multicast

SRTP: Selectively Reliable Transmission Protocol

S-SCTP: Smoothed SCTP

SWS: Shared Window System

TCP: Transmission Control Protocol

TOAST: Test and Optimization of Applications in Surgery and Training

TTL: Time-To-Live

UDP: User Datagram Protocol

UI: User Interface

VRML: Virtual Reality Modeling Language

Chapter 1: Background and Motivation of Tele-Haptic Class of Applications

1.1 Introduction

Haptic Collaborative Virtual Environments, like other collaborative environments, have become a very popular research area for many years through networks. With the growth of the computer network, internet and intranet technology, more and more people would like to work together from places that are geographically diverse. People need a shared workspace similar to physical working environments in the real world in order to participate in a simulation to manipulate shared objects. Collaborative Virtual Environments (CVE) are such an approach. CVEs share virtual reality spaces in which geographically separated users can participate in a simulation to manipulate shared objects. In addition, CVEs also allow participants to collaborate in closely coupled and highly synchronized tasks. Moreover, when used with Haptics, CVEs have been shown to further improve the quality of such object manipulations because of the Haptics provision for the sense of "touch" and the force-feedback capability [23]. These tasks require enhanced communication and collaboration between distributed users, such as two remote engineers practicing the fixing of a space shuttle, or a tele-surgery training application. Such environments play a tremendous role in military training, tele-presence, collaborative design and engineering, entertainment, and many other personal and industrial applications [25]. In a CVE, the shared space is dynamic and interactive, changing over time based on the participants' actions as well as other factors.

One of the problems in such environments, which has been studied and addressed to a certain extent in recent years, is network lag. Shared object manipulation is achieved in CVEs by sending each user's interaction with the object to other participants over the network [36]. Because of network limitations and traffic conditions, some of these interaction updates are lost or delayed. In fact, network lag is present in any networked distributed application; Tele-haptic CVEs are no exception. Due to its requirements for closely-coupled tasks, synchronized collaboration is especially susceptible to delay and

jitter. Over the past few years, many studies to examine the effect of delay and jitter on synchronized tasks performed by users in such environments have been conducted [44] [18]. It is generally agreed that in order to perform closely-coupled tasks in CVEs, an end-to-end delay of no more than 200 msec is required [31]. In addition, it has been shown that jitter, which in simple terms can be thought of as the variation of delay, has a significant detrimental effect on the quality of a collaborative session, with a low 10 msec jitter resulting in a collaboration environment. This is almost as bad as one with a 200 msec delay but no jitter [31]. A great deal of research has been done to compensate for network lag in order to provide a better quality of service for collaborative virtual environments. Some of these studies concentrate on network loss [37], while others try to address the jitter problem [10]. These approaches, however, focus on solutions based on network communications either at the transport-layer, the network-layer, or the application-layer in the form of framing of update messages.

Recently, researchers have investigated two other approaches. One is the use of *decorators* [16] – visual cues that inform the user, ideally in a non-intrusive manner, about the state of network lag. For example, the color of the pointing cursor changes depending on the degree of network lag. In contrast to the networking-solution techniques discussed above, this approach does not try to hide network lag from the user but in fact involves the user in the process. By seeing these visual cues, users are aware of the state of delay in the application and can cope with it using intuitive strategies such as slowing down or waiting longer for the other side if possible.

The other approach, which has been used for some time now, is prediction. Dead reckoning techniques, for example, which have been used in distributed military simulations for a number of years, can predict an object's location and orientation using its last transmitted velocity and timestamp [20]. However, it is a known fact that for closely-coupled tasks that are highly synchronous, dead reckoning does not perform well [38]. Dead reckoning or prediction is now being re-examined by researchers; prediction in collaborative environments [11, 15] has recently been revisited with some positive results.

In this thesis, all of the above approaches are applied to a Haptic CVE environment in which two users must finish a given task collaboratively. To the best of

my knowledge, no other work has attempted to combine these various types of solutions in the same application and study and report the effect of such a hybrid solution. My contribution lies in the design and architecture of the hybrid approach, specifically for haptic collaboration, and in the objective demonstration of the superiority of the combined solution over a single approach.

1.2 Existing Problems in Tele-Haptic Based Type of Applications

Many haptic incorporated collaboration approaches have been developed in the last few years. However, these techniques cannot completely remedy the problem. In addition, a certain lack of synchronization is always experienced between remote users. Even in the presence of such approaches, several shortcomings have been discovered during the use of these systems. In most existing systems, there remain severe difficulties in developing environments that satisfactorily support interaction between individuals within a virtual world. In particular, the current haptic, tactile, and olfactory interfaces are extremely unsophisticated, and more haptic research is being conducted in order to make it more sensible. In the real world, we feel a physical sensation when someone takes, moves, or deform an object in our hand, even if we are not looking at it at the time. However, even though many techniques have been adopted, the current state of CVEs does not attain such a reality. Applications requiring haptic and tactile feedback – the sense of physical force for training – are some of the main drivers for producing haptic feedback in Virtual Reality systems. Touch is key, too. For example, touch is important in training for medical skills or in feeling an object. Given the usual limited interface's field-of-view, it is quite possible that a user is not able to see an object that is being taken from him or her. As a result, because the user lacks haptic cues that enable him or her to notice this incursion, he or she is puzzled upon referring back to the object in his or her possession to discover that it is not there. Moreover, in the real world, both participants have human perceptual resources, such as a user's view of another's action. The field of view in a desktop interface in a CVE application makes user encounter problems in discerning what fractions of the virtual environment other users are able to see when discussing and referring to features of the virtual world.

Even though the realism of graphics technologies is rapidly advancing, interfaces to real-time expression for such graphical illustration lag far behind. Typically, a graphics application will refresh the contents of the "framebuffer" approximately 30-60 times a second in order to give the human eye the impression of continuous motion on the screen. However, a haptic application will refresh the forces rendered by the haptic devices approximately 1000 times a second [33, 34] in order to give the kinesthetic sense of stiff contact. If the frame rate of a graphics application is run at a rate lower than 30 Hz, the user may perceive discontinuities in an animation such that the animation no longer appears visually smooth. Similarly, the user may perceive force discontinuities and a loss of fidelity when the haptic device is refreshed at a rate below 1000 Hz. As a result, haptics and graphics rendering are typically performed concurrently in separate threads so that each rendering loop can run at its respective refresh rate.

Users might be standing next to one another, interacting or manipulating the same object concurrently in the virtual environment, perhaps even collaborating with other users in order to work with the same virtual object, but may be physically located in different places or on opposite sides of the planet. In this case, network delay will be far greater due to network hardware and current congestion problems. Such delays make it extremely difficult to create the illusion of both users instantaneously interacting with the object in a perfectly consistent way.

In this thesis, four network parameters are considered:

- *Delay:* This impairment exists in any computer network since electrical and optical signals have a finite propagation speed. In addition, processing information consumes time at both end points. The collective effects of transmission delay, switching delay, queuing delay, retransmission delay, and processing delays varies greatly according to the type of network involved, changing network conditions, and processors (sender, receiver, or servers). In haptics and collaborative virtual environments, this creates detrimental effects to successful collaboration, a few of which are highlighted above.
- *Jitter:* This is the variance in the end-to-end delay among distributed users. Studies have shown that a small amount of jitter impedes collaboration by as

much as a big delay. Collaboration can become as difficult over a network whose delay varies between 0 to 20 milliseconds as over another network with a fixed 200-millisecond delay. Although techniques have been proposed for reducing jitter by buffering [10] the update messages at the receiving side or compensating for jitter by using visual cues [37] that indicate network lag to the user, these methods are inadequate.

- *Packet loss*: This occurs when networks discard data packets when the network is overloaded and cannot accept any incoming data at a given moment. TCP/IP ensures the delivery of all data sent in a transmission. It verifies that all data is received properly at the other end. If there is any failure in the delivery of data upon first try, it will request that the data be resent to the receiving device. The network may be unreliable and drop packets.
- *Bandwidth requirement*: The synchronous collaboration transport protocol (SCTP), which is a host-to-host layer protocol and is mapped into a communication protocol in this technique, sends critical update messages reliably and normal update messages with best effort. SCTP is encapsulated into UDP packets and assumes that the underlying physical network supports IP multicasting. Each update message, which represents only one packet, contains 48 bytes plus the protocols overhead to transmit six double-precision floating-point numbers, which makes 90 bytes if UDP/IP is used over Ethernet. Sometimes, this might appear to be a small size. However, a haptic application will refresh the forces rendered by the haptic devices approximately 1000 times a second, which is a much larger update rate than the update rate of video. If each update requires 90 bytes and is transmitted every millisecond (i.e. 1000 times a second), the bandwidth requirement becomes 90 Kbytes/s, which is also an important factor to consider. Moreover, if the critical updates named as key update messages fail to be acknowledged before the timeout occurs, the same update is sent again as a key. Thus, the acknowledgement packets in SCTP create a great deal of traffic that affects the network bandwidth.

In terms of communication protocols such as SCTP, Smooth-SCTP, or UDP that have been proposed for collaboration virtual environments in literature, the focus of this thesis has been lain on the synchronous collaboration transport protocol and the Smooth-SCTP, which implement reliability for key update messages and ensure that these key update messages are received on time. In fact, SCTP ensures the transport of haptics update messages among users and the prediction method can generate lost or delayed update messages hence, network traffic is reduced substantially. Another protocol, Light TCP, which buffers update messages (an updateable queue) on the sender side, has not been considered here because it does not meet collaboration requirements. [10]

1.3 Thesis Objective and Contributions

As demands for tele-collaboration increase, the requirements of collaboration systems have also increased. The main goal of tele-collaboration is to provide haptic development architecture independent from the application so that the network transport protocol does not need to be reinvented each time a new distributed application that uses haptics is developed.

A contribution to this thesis has been made by combining the two techniques used to moderate the simulation lag in haptic-based virtual environments and to increase awareness of the state of the delay in the application and cope with it using intuitive strategies. A decorator, which is a visual cue embedded in the haptic virtual object, is used to inform the user about the state of the simulation lag. In addition, an algorithm that predicts users' most likely actions in the very short term and compensates for delayed or lost packets by interpolating collaborative actions controls this decorator. In other words, this technique improvises the current network delay/loss and indicates it in a non-intrusive manner to the local user. To accomplish this, we have:

- Studied the CVE applications' particular needs in terms of communications;
- Studied haptics and its interaction with virtual objects over networks;
- Designed and implemented suitable prediction models;
- Designed and implemented suitable protocols;
- Implemented a decorator that changes color with the variation of network delay;

- Implemented a testbed for evaluation;
- Implemented two applications using haptics in collaborative virtual environments.

However, we have found during tele-haptic class of applications that

- In the presence of network delay, packet loss and jitter-prediction of the lost update at the receiver side helps collaboration.
- Because of the lack of view of individual participants' activities (awareness), and haptic feedback, the decorator reveals drawbacks in front of participants and helps with collaboration.
- Decorators reveal the corrections due to collaboration drawbacks, distinguish the discontinuities from participant-interacted actions, and restore the interaction for the participant after a discontinuity or correction.
- Prediction capabilities and indication of level of trust associated with the prediction, which helps the users in anticipating impending states of the system. Such as the real position or movement of a virtual object.
- Reduce network traffic because some state update messages send reliably and some regular state update messages send best effort transport.

Based on these findings, a prediction model that extends the synchronous collaboration transport protocol by creating a history – buffering the update messages once they are received at the receiving side – and a predictor – to predict the future and in-between update messages at the receiving side so that in prediction a processing delay is experienced – is proposed.

1.4 Thesis Organization

This thesis describes special techniques and solutions required in order to provide a better quality of service for collaborative virtual environments and help users collaborate in a more natural way in spite of network lag, limited fields of view, and lack of haptic feedback. The goal of the research is to build such a system, which combines both

network-level approaches and user-interaction-based approaches that support tightly coupled collaborative tasks to be performed efficiently in virtual environments. The remainder of the thesis is divided as follows:

- Chapter 2 presents the basic concepts that will be used throughout the thesis, including background knowledge concerning haptic incorporated CVEs, a description of certain haptics CVE technologies, a statement of 3D graphics technologies and the network issues that arise in collaborative virtual environments.

- Chapter 3 presents related works about the collaborative tele-haptic application. In addition, a number of similar works done by other teams are introduced.

- In Chapter 4, the proposed architecture is presented. The architecture deals with interaction stream, abstraction of update messages, linear prediction model and decorator design. Based on this architecture, two tele-haptic CVE applications were designed and implemented which will be explaining in chapter six.

- Chapter 5, it is presented Experimental environment, which is providing an evaluating test bed or the prediction algorithm and transport protocol. Detail analysis of Adaptive-Transport layer Optimization for Applications in Surgery and Training (ATOAST) is presented.
- Chapter 6 covers the two applications and test results. This chapter includes measurement and evaluation of the tele-haptic based application with or without prediction algorithm and decorators.

- Chapter 7, a conclusion is given and some future work is mentioned.

1.5 Publications Resulting From This Research

1. A. Boukerche, S. Shirmohammadi, A. Hossain "Prediction Based Decorators for Distributed Collaborative Haptic Virtual Environments", *J. Computer Applications in Technology, Special Issue on Collaborative Multimedia Applications in Technology* (accepted, to appear), 2006.

2. A. Boukerche, S. Shirmohammadi, A. Hossain "Moderating Simulation Lag in Haptic Virtual Environments". *Proc. 39th Annual Simulation Symposium,* Huntsville, AL, USA, 2006, pp 269-276

3. A. Boukerche, S. Shirmohammadi, A. Hossain "The Effect of Prediction on Collaborative Haptic Applications", *Proc. IEEE Virtual Reality Conference*, Alexandria, VA, USA, 2006

4. A. Boukerche, S. Shirmohammadi, A. Hossain "A Prediction Algorithm for Haptic Collaboration". *Proc. IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE 2005),* Ottawa, Ontario, Canada, 2005. pp 154-158

Chapter 2 Tele-Haptic Incorporated CVE

In this section, the basic concepts that will be used throughout the thesis are introduced. Examples and information concerning how virtual environments are currently being used, how haptics are incorporated, and distributed users collaborated, are presented. Virtual environment research is also shortly reviewed.

2.1 General Concepts

Virtual environments can be used to run simulations of various applications such as medical training, surgery planning, tele-mentoring, industrial training, and gaming where geographically-distributed users collaborate in real-time, sometimes in a closely coupled fashion, by manipulating shared objects. Such environments are also known as Collaborative Virtual Environments (CVEs). Several different definitions of the term virtual environment (VE) and networked virtual environment exist in the field of literature. The most compelling and heralded definitions are given below.

"*The screen is a window through which one sees a virtual world. The challenge is to make that world look real, act real, sound real, feel real.*" (Sutherland) [40]

"*Virtual Reality creates a new objective level of reality. If you're ever confused about which reality you're in, you put your hands on your eyes and see if you're wearing Eye-phones or not.*"(Lanier)[24]

"*Networked virtual environment is a software system in which multiple users interact with each other in real-time, even though those users may be located around the world.*" (Zyda and Singhal) [38]

A Virtual Environment (VE) is the nonphysical, digital, and machine-readable representation of an environment, and evolves over time as a result of both timed events

and user input. If the virtual environment is a moving ball, the ball will pass distance or change speed with time (it is time-dependent) and stop, change direction, or speed by users' interactions.

A networked VE means that the user interface to the networked VE updates its information more than 10 times per second. The user interface is often a view of the environment. The general aim of a networked VE is to create an immersive experience via realistic three dimension graphics, stereo sound, and interaction. Going deeper into the ideology, Zyda and Singhal [38] enumerate five following common features that could be used to distinguish networked virtual environments.

- *A shared sense of space* – The three-dimension representation of shared virtual places allows the participants to have an illusion of being located in the same space. The environment of such space must represent the same characteristics such as acoustics, weather, temperature, etc., to all participants.
- *A shared sense of presence* – By virtue of virtual persona represented as avatar, each participant has a virtual persona in the shared environments so that participants able to enjoy a shared sense of presence. The avatar has a graphical representation in the virtual environment as well as a physical model, a motion model, and other characteristics. A spacecraft, a warrior, a tank, an airplane, a ship, or a human model is examples of possible avatars.
- *A shared sense of time* – A participants locates him or her in shared virtual environments and interacts with another participant in real-time via the communication network. A participant able to see each other's actions as he or she is occurring, allowing real-time interaction that he or she experiences a shared sense of time.
- *Ways to communicate* – In such environments, participants have at their disposal ways of communicating with one another: by typing text, speaking, or gesturing using their avatars. Most networked training systems and games enable some type of communication between users in addition to them merely seeing each others' actions. This communication might occur by talking into a microphone, which the others in the environment hear if their avatars are close, or by typing text, etc.

- *Ways to share common objects*– Participants have a way to share common objects in virtual environments: either passively, by objects that provide a common frame of reference for experience, or actively, by objects with which users can directly interact. In more complicated networked virtual environments, users can interact with the virtual environment by altering it and perhaps using and picking up objects. They might also be able to give these objects to each other or use something – a car, for instance – together with the environment.

Collaborative Virtual Environments (CVE) are evolutions of the networked VEs. A virtual environment is said to be collaborative if it is networked, and if users are able to collaborate, to interact with each other in real time. Such a thing may seem obvious in the real world, but becomes an issue as soon as a computer network is involved. When used with Haptics, CVE have been shown to further improve the quality of such object manipulations because of Haptics' provision for the sense of "touch" and the force-feedback capability [19, 23]. Considering the degree of synchronization or level of realism when, for example, two remote engineers practice the fixing of a space shuttle part, or a remote surgeon teaches a specific operation to an intern, it is clear that CVEs put stringent requirements on the application in terms of ensuring all users have the same view of the shared objects with the smallest skew possible.

2.2 Input and Output Devices for Virtual Environments

Virtual environments usually are, or try to be, immersive, i.e., they make the user "step inside" the application. This trend has provoked the development of new devices that offer a more sensitive, more instinctive and more realistic way to exchange information with the computer than the mouse, keyboard or monitor. Such devices include:

- *Head-mounted displays*: A head-mounted display (HMD) is a display device that users wears on the head to have video information directly displayed in front of the eyes. It has either one or two small high-resolution LCD or OLED monitors with magnifying lenses embedded in a helmet, glasses or visor. HMD are placed

in front of the user's eyes, and display 3D stereoscopic images by displaying an offset image to each eye.

- *Polarized goggles*: The interaction of the polarized filters with colors or various characteristics of the likeness on the screen shifts the visions and create 3D graphics, usually on a wide screen or a set of screens;

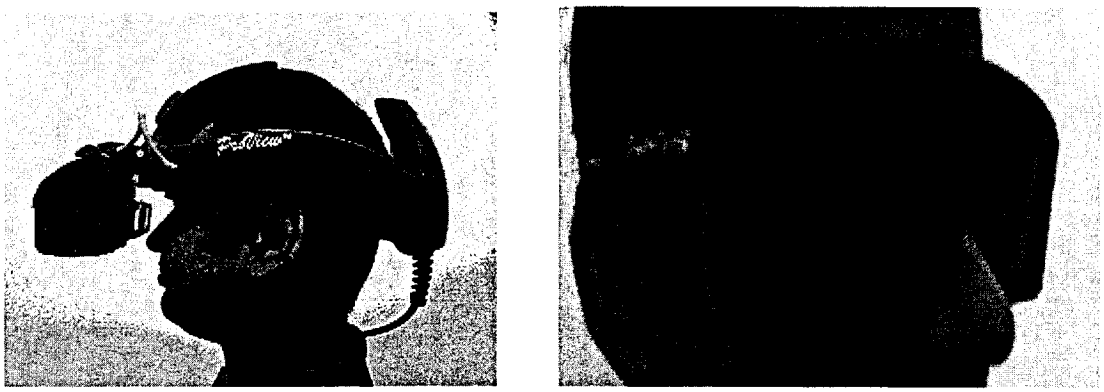


Figure 2.1: Head-mounted Display (HMD) and Polarized Goggles.

- *Dataglove and trackers*: A *dataglove* is a glove-like input device for virtual reality environment. Various sensor technologies are used to capture physical data such as bending of fingers. A tracker is attached to capture the global position or rotation data of the glove. Dataglove and tracker capture the movement of the body in the space. If trackers register the movement of the user's head, the picture shown by a head-mounted display can be updated accordingly. Several haptic gloves are designed for virtual reality simulations – "Rutgers Master", the LRP Glove, and the "CyberGrasp" etc. Force feedback bandwidth for these devices is in the range of 10 – 50 Hz. These gloves have one or more force degrees of freedom (DOF) per finger with forces grounded in the palm or on the back of the hand. A virtual hand maps the user's hand to the virtual environments.

- *Haptic devices*: Haptic interfaces the users via the sense of touch by providing the force feedback. The term "haptic" comes from the Greek Hapthai. It means "pertaining to the sense of touch". Virtual objects can be felt and manipulated

through haptic devices. It is an emerging technology that promises to have wide reaching implications.

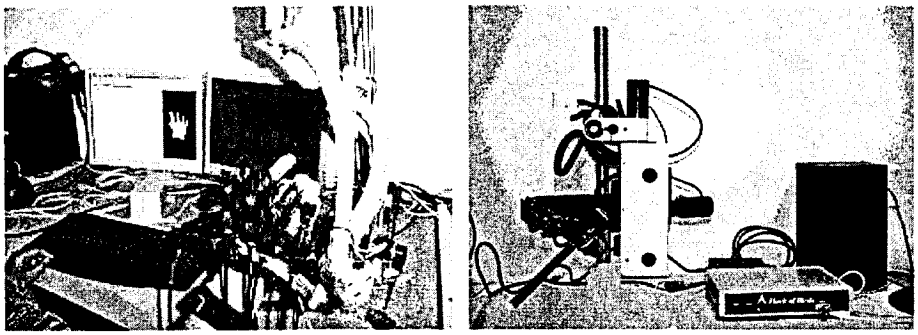


Figure 2.2: Dataglove and Trackers and Freedom force feedback haptic device.

Haptic devices give users a sense of touch with computer-generated environments by force feedback, so that when virtual objects are touched, they seem real and tangible. Visualize holding a stick or a ball and moving it through this air: no force is felt except the gravity. If the stick or the ball touches a brick-wall, it will not be able to go through it because of the reaction force the brick-wall exerts on the stick or the ball. No matter how hard one pushes the stick or the ball; the stick or the ball pushes the manipulating hand back. Again, consider a medical training simulator in which a doctor can feel a scalpel cut through virtual skin, feel a needle push through virtual tissue, or feel a virtual bone. Haptic devices simulate such behavior by having the stick or scalpel sending a force to the hand, as if it was touching something they can also simulate soft structures (imagine the feeling the user would get by pushing a stick against a snow-ball or a surgeon cut a virtual organ), or even different roughness of surfaces (to distinguish, for example, silk from sandpaper, or muscle from bone). Haptic devices function as input devices (they can send their position and the force applied by the user to a computer), at the same time as output devices, the output being the force (and, depending on the device used, torque) generated. In this thesis, I use the PHANTOM and describe later.

2.3 Using Haptics in Virtual Environments

Haptics is gaining widespread acceptance as a key part of Virtual Reality (VR) systems, adding the sense of touch to what were previously visual-only solutions. The major areas

that use virtual environments are visualization, simulation and training, and teaching and entertainment. With the progress of technologies, more virtual environments work hand in hand with virtual reality because of the obvious need for virtual environments to be immersively visualized. Whatever uses virtual reality also uses virtual environments. Virtual reality applications are a subclass of virtual environment applications. All virtual reality evolves from virtual environments, but all virtual environments do not spawn from virtual reality. In other words, it is possible to have virtual environments without virtual reality, but the opposite does not always work. Virtual environments equipped with haptic rendering ability allow users not only to see virtual environments, but also to interact with them. The haptic force, or moment display, for virtual environments is composed of a force or moment interface for displaying realistic sensations, and a visual interface for displaying virtual senses. In this system, users will feel as though they are actually manipulating real objects by holding effectors, which are shown in Figures 2.9, 2.10 and 2.11. Here I present simple examples of how VEs are used in the categories and applications that have been mentioned, with haptics creating more realistic solutions. The areas in which networked virtual environments are used will be mentioned as well.

2.3.1 Visualization

The most important use of haptic interfaces with collaborative virtual environments is data visualization. A variety of objects can be translated into touchable forms, whether they are three-dimensional plots of mathematical functions, the Earth's atmosphere, part of a car engine, the architectural model of a high-rise building, or a prototype statue. Many areas of sciences, including architecture, engineering, and information visualization, use virtual environments and virtual reality. Usually the motivation of such applications is to represent a large amount of information in a clear, understandable way. Currently, architects are collaborating with other architects in order to construct buildings in an entirely new way: through computer networks. They connect through computer networks to design buildings by constructing them in a virtual environment. In order to accomplish this, architects use generic virtual reality systems, mostly for customer presentation. Virtual reality allows people to walk inside the planned building and get a solid idea of what the building will look like before the construction decision is made. If

the customer wishes to make changes, he or she can input them in the model. And if the virtual reality architecture software is sufficiently good, the designer can even modify the building interactively when he sees something he wishes to alter as he moves through the model. Being able to walk inside the virtual representation of the building is an extremely powerful and effective customer presentation tool.

Architects and scientists are not the only ones who can take advantage of this technology. Engineers, scientists, surgeons, and trainers can also represents their work in virtual environments and modify what they create by collaborating with others. At the moment, engineering design is done mostly with the use of computers. All kinds of parts, from knots to full complex machines, are constructed using Computer Aided Design (CAD) programs. Large manufacturing companies are equipped with machine-software infrastructure that makes it possible to feed virtual CAD designs into their manufacturing systems, which then construct a real world object based on the directions they receive from the virtual model.

Haptics can also be useful in the education sector, and not only for able-bodied students, but also for the blind or people who have only partial vision. With haptics, these people will be able to feel a line on a two-dimensional plot as easily as an able-bodied person is able to see one. Virtual sculpture is designed to be touched, but unlike its real world counterparts, it will never break or prove to be inaccessible. Normal artists' techniques can be translated into virtual reality environments so that people who are not artistically inclined can learn how to properly hold a virtual paint brush and feel how an artist would have used it to paint.

A haptic interface was recently connected to an atomic force microscope that showed the placement of atoms and molecules within crystals so that the 'surface' texture of the rows of atoms could be felt. With the help of haptics technology, this system could be adapted so that individual atoms could be manipulated merely by picking them up, thereby allowing for nanoscale devices to be hand built by the operator.

2.3.2 Medical Sector

The most common types of data acquisition for medical imaging are based on sophisticated imaging methods, including Magnetic Resonance Imaging (MRI), and

Positron Emission Tomography (PET). Using these programs, medical personnel are able to construct three-dimensional virtual models of all kinds of things. For example, they can create the representation of a patient's actual skull bone or any other body part. These imaging methods can also be used within other medical contexts. For instance, they can be used to determine whether a patient has a disease or tumor, eliminating the need for surgery. A segmentation technique is used to convert two-dimensional layers into a three-dimensional volume model. Both CT and MRI image processing use the segmentation method to produce volumetric models, which can be seen as three-dimensional rectilinear grids of volume elements, also known as voxels. Models that have been constructed can then be used to see exactly where the tumor or other medical problem is situated. The models can then be of significant help in planning and executing the surgery required. Using this technology, it is even possible to project the three-dimensional model on the real head of the patient, in true scale, so that the surgeon is able to see under each layer of tissue and avoid damaging the areas that must not be damaged. If the surgery is too complicated, the surgeon can collaborate with other surgeons so that he or she can make himself or herself even more skilled. This kind of visual mixing of real and generated imagery, and virtual and real environments, is known as augmented reality.

2.3.2.1 Tele-Surgery

Various haptic interfaces that have been designed for medical simulation may prove to be especially useful for training surgeons to conduct minimally invasive procedures such as laparoscopies and interventional radiology, and remote surgery using tele-operators. In the future, expert surgeons may work from a central workstation, performing operations in various locations, with machine setup and patient preparation being performed by local nursing staff. Instead of traveling to an operating room, as is the current practice, the surgeon becomes a tele-presence. One of the particular advantages of this type of work is that the surgeon is able to perform a great deal more operations of a similar type than he is used to, all while experiencing less fatigue. It is well known that the patients of a surgeon who performs a higher number of procedures of a given kind will have statistically better outcomes.

In ophthalmology, "haptic" designates a supporting spring. Two such springs hold an artificial lens within the lens capsule after the surgical removal of cataracts. [12] Deals with a hapto-visual application that illustrates eye cataract surgery performed on a human and is a specific research topic of Distributed and Collaborative Virtual Environments Research Laboratory (DISCOVER Lab) at the University of Ottawa. What is of particular significance is that this surgery used haptics to bring together geographically distant surgeons who interact in real-time. Haptic devices can be found at both ends, with an instructor tele-mentor found in one location and a trainee at the other.

Robotics and haptics have allowed for another very exciting application for virtual environments to come into existence. This application is telesurgery [1]. If a surgeon is to perform a surgery or other action on patient who is at another location, it is absolutely essential that he obtain force feedback. Thus, there must be cameras to capture and transmit what the surgeon should feel while he is performing his actions, and also to show the patient's reactions. For this reason, the input device manipulated by the surgeon must be haptic; otherwise, it would not be able to copy the reactions felt by the robotic device that acts on the patient.

Because of their nature, certain surgeries are easier to transpose in a virtual environment than others. These are minimally invasive surgeries. In these types of surgeries, a natural orifice of the body or a small incision of less than one centimeter is used to insert several tools and a mini-camera inside the body. These tools and mini-camera enter the operation site without directly exposing the latter through a large incision. This is done in many minimally invasive procedures such as laparoscopy and intravascular surgery [36]. These techniques have been developed because they are significantly less invasive than traditional surgery. The drawback, however, is that the surgeon cannot operate his or her tools directly, but must instead manipulate them through handles. These handless are connected to the tubes inserted in the patient's body; the tubes themselves contain the tools. Given that at this point the surgeon has already lost "direct contact" with the patient, it is not very difficult, conceptually speaking, to have him or her operate such tools remotely.

The first instance of tele-surgery took place in 1993. Since then, a number of successful surgeries have been performed. At times, the surgeon and patient have been physically separated by as much space as one ocean.

Another development of the practice of tele-surgery is surgery simulation and remote training. This thesis presents a tracheostomy training application in which surgeries have been simulated between two geographically distant locations such as Canada and France. Other research efforts explored here include the modeling of human tissues and tool actions, the tools being scalpels or surgical hooks.

2.3.3 Simulation and Training

Haptics, the science of touch, is being applied in virtual reality environments to increase realism. Haptics has been applied very recently to education, training and simulations purpose, most notably in medical education and distant learning procedure. In the Stanford Visible Female project, a three dimension stereoscopic visualization of the female pelvis has been developed from numerous slices of two dimension pelvis data. A virtual reality-based simulator prototype for the diagnosis of prostate cancer has been developed using the PHANToM [27] haptic interface. Simulated automobile steering wheels are now available which provide road feelings for racecar simulations.

Simulation training centers use virtual environments quite extensively. For example, military air forces and commercial airlines train pilot cadets on simulations as opposed to real planes for a long time before the cadets actually get to fly a real plane. The operational costs of VEs are much lower than the costs of using the real objects and environments that they simulate. The United States military uses simulations in every part of its training. The army, troops, commands, the navy and the air force all employ simulations. The military is constructing its systems so that it can obtain increasingly large networked joint simulation capabilities. The goal of this large networked joint simulation is to create networked virtual environment in which entire war operations can be simulated using thousands of soldiers, troops, tank drivers, and pilots interacting together. Being able to practice an operation in a simulated environment before the real event helps a great deal when it comes to dealing with such situations. It is an essential service. If one would like to see the impact of these projects illustrated in greater detail,

one can consult the United States Army's simulation instance website, where army simulation projects can be found [41].

Nowadays, astronaut of spacecraft, captains of commercial airplanes, navy captains, and drivers of any type of complicated and expensive machinery are trained, at least some time during their training, in a simulated environment. Simulation training is especially convenient when it comes to practicing uncommon and surprising situations, such as emergencies, that cannot be constructed in the real world. VEs can be used to obtain an authentic feeling and a degree of realism that is otherwise impossible; thereby helping people using the systems visualize the situations in which they have been placed.

2.3.4 Entertainment

The use of virtual environments within the entertainment sector is increasing at an enormous speed. The entertainment industry is quickly emerging as the largest individual user of virtual environments and real-time visualization. It has also recently become the main technology driver of the sector. Some low-end haptic devices are already common. Joysticks and game controllers that vibrate upon command from the game software provide haptic feedback.

Entertainment products are found in three large sectors: movies, computer games, and music. In this day and age, the budgets of major movies can be over a hundred million dollars. The budgets of computer games have also increased rapidly with today's quickly-evolving computer hardware. Fifteen years ago, a computer game consisted of a couple of megabytes of software, images, models, and sound. Today, however, games easily contain few gigabytes' memory of these elements. Of course, the image and sound quality of such games has improved, but so has the size and complexity. Nowadays, creating a game requires as large a number of people as are used for making movies. And while a few short years ago games were not advertised on TV or during cinema movie previews, this practice is very common today. Even though the budgets of blockbuster movies are still more than the budgets of big games, a few years ago the game industry passed the movie industry in annual turnover; it is currently dealing with more money than the entire movie industry.

2.4 Networking and Collaborative Virtual Environments

Networking makes it possible for distributed users to exchange information. However, users may be located in different places physically or else on opposite sides of the Earth. During the past few years, computer network technology such as data processing, routing, and transmitting has grown at an extremely fast rate, so much so that the world has become a global village that expands over most of the planet. The Internet is made up of an incredible amount of computer networks, which act as the medium through which computers can exchange information, and routers, which forward data packets along networks.

Because of the nature of the Internet and its users, there must naturally be standards concerning the manner in which information is exchanged and presented over this medium. Information may be required to travel from the other side of the world through several different networks and routers. All of these networks and routers must be able to process the form in which the information is presented. The standards that specify how information is exchanged in networks are known as protocols. A large part of this project involved networking simulation environment components together. The first objective was to choose a networking approach.

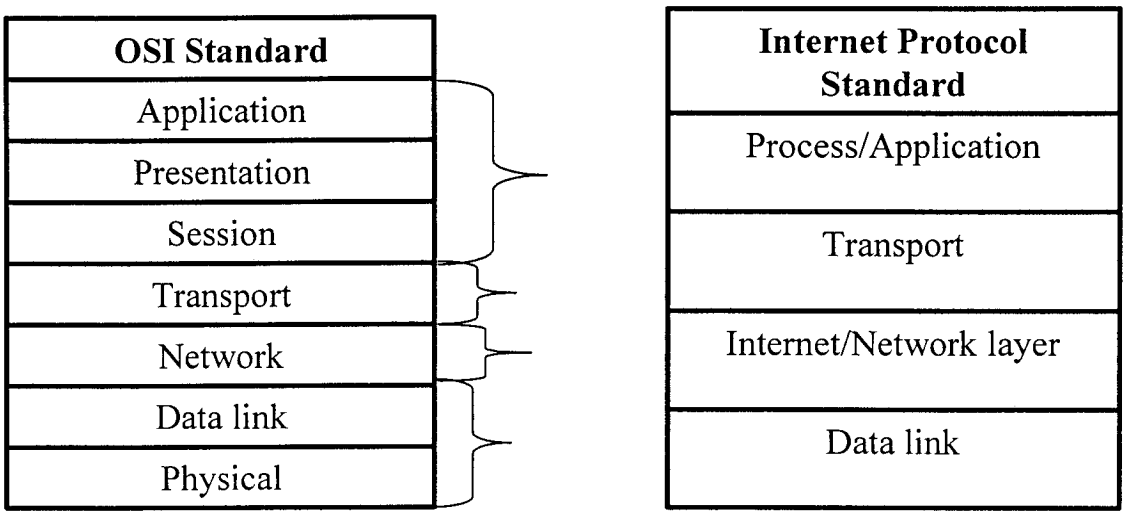


Figure 2.3: OSI and IP protocol layer models, and Message passed between corresponding host layers.

Process/Application	The upper layers named as Application layers deal with application issues and usually are implemented only in software. The applications invoke TCP/IP services, send and receive messages, or stream with other hosts. Delivery of update messages can be intermittent or continuous.
Transport	This layer provides end-to-end communication between applications using either reliable delivery connection-oriented TCP or unreliable delivery connectionless UDP in packetized form. Transfers packets end-to-end with other hosts. This layer is responsible for end-to-end error recovery and flow control.
Internet/Network layer	Provides switching and routing technologies that encapsulates packets with an IP datagram that contains routing information, and receives or ignores incoming datagrams, as appropriate, from other hosts. Checks datagram validity, handles network error, packet sequence and congestion controls.
Data link	Encodes and decodes data packets into bits. Includes physical media signaling and lowest level hardware functions, and exchanges network-specific data frames with other devices. Includes the capability to screen multicast packets by port number at the hardware level. It also includes the Media Access Control (MAC) layer and the Logical Link Control (LLC) Layer.

Figure 2.4: Summery of Internet Protocol (IP) suite layer functionality.

Collaborative Virtual Environments (CVE) are evolutions of the networked virtual environments and allow the participants to collaborate in closely coupled and highly synchronized tasks. These tasks require very close coordination between two or more participants using an end-to-end communication architecture can be used for tight coupling and real-time response of physics-based models. The first step to build a CVE is to implement a networked virtual environment. The integration of networks with virtual environments occurs by invoking underlying network functions from within applications. Figure 2.3 shows how the seven layers of the well-known Open System Interconnection (OSI) standard network model generally correspond to the effective layer of the Internet

Protocol (IP) standard. Functional characteristic definitions of the IP layers follow in Figure 2.4.

In order to be able to understand the process of networking virtual environments, we must first have a basic knowledge of networks, communication requirements, and network architectures. Knowledge of these elements helps one understand protocols and their properties. Before we discuss different protocols and networking libraries, however, we will review several design issues, which have to be faced in haptics, induced collaborative virtual environments.

2.4.1 Network Architectures and Data Management

The information exchange currently used by the Internet is based on packets. No matter what kind of information is being sent, it must first be divided into packets. Each packet contains two parts: a header and a message. The header defines the packet's target address as well as some other processing information. A thorough look at the different types of network architecture is still required before protocols can be discussed. In this chapter, the term "network architecture" refers to the connection layout of virtual environment networks.

The most common architecture that has been used in network games and virtual environments in the past is client-server architecture. Client-server structure means that there exists a main node to which all other nodes connect. The main node takes care of routing information to the right clients. It might also process some of the global functions of the world. All of the CVE elements are stored in the server; when an entity moves through the virtual environment, the server updates the state of the object upon the client's request. When a client needs to process a virtual object in CVE, he or she downloads the object from the server, processes the object, and informs the server. Therefore, the server handles the site management and knows where every client is situated in the virtual world. However, in a large and dynamic collaborative virtual world, servers can quickly become I/O bottlenecks with increasing network delays, latency, packet loss, and jitter.

In the peer-to-peer structure, communication goes directly from one participant to another. This structure is preferable to other client-server approaches in this day an age as

long as smart ways of routing information are used. The peer-to-peer structure has a homogeneous fully-replicated distributed CVE. All of the objects of the CVE are dynamic and use reliable multicast protocols to actively replicate new objects. No central management system exists in the virtual universe and every participant is therefore responsible for ensuring that the global description is accurate. In client-server architecture, the server will bottleneck when the amount of nodes in the network becomes overly large. Experience and research have shown that even in the case of networks with a high processing capability, bottlenecks can occur. Client-server networks also contain more latency than peer-to-peer networks because packets travel through indirect paths in this type of network, first from each node to the server node and then from the server node to other nodes. The client-server structure is also a structure with a single point of failure. If the server crashes, the entire system becomes unusable. Peer-to-peer systems are naturally more problem-tolerant. On the other hand, peer-to-peer architecture is vulnerable in terms of large scalability because of the costs of communication associated with maintaining reliability and data consistency across wide area networks.

2.4.2 The Internet Protocol

Internet protocols are the world's most popular open-system protocol suite because almost the entire Internet, including LAN and WAN, uses Internet Protocol (IP) to communicate. The two best-known protocols are the Transmission Control Protocol (TCP) and Internet Protocol (IP). The Internet protocol suite is a packet-based low-level protocol (such as TCP and IP) that also specifies common applications such as electronic mail, terminal emulation, and file transfer. Hosts and routers ensure that packets transfer from a source host to a destination host. Internet Protocol (IP) is a network-layer protocol that contains the addressing information and some control information that enables packets to be routed. Along with Transmission Control Protocol (TCP), IP represents the heart of internet protocols. IP hides the properties of the transmission line it uses to travel on. It includes facilities to split the packets into smaller fragments when traveling through networks that do not support large packages, and facilities to reassemble the packages at the other end. To avoid the possibility of routing the packages in infinite loops, IP has a time to live field in the packet's header section. At each router, the time to live field is

decreased. If it reaches zero, the package is destroyed. Thus, IP provides connectionless, best-effort delivery of datagrams through an internetwork and provides fragmentation and reassembly of datagrams to support data links using different maximum-transmission unit sizes. IP is an old, standardize-based protocol in networking. IP supports almost all other protocols, which are built on top of IP and use it as their lower-level implementation.

2.4.3 TCP

TCP stands for Transmission Control Protocol, which is layered on top of the lower-level Internet Protocol. TCP provides reliable transmission of data in an IP environment and corresponds to the transport layer of the OSI reference model. TCP offers reliability by providing connection-oriented, end-to-end reliable packet delivery through the internet. It does this by sequencing bytes with a forwarding acknowledgement number that indicates the next byte the source expects to deliver to the destination. It automatically transmits acknowledgements of actions. TCP notices if a packet is lost and asks that it be retransmitted. TCP also embodies data verification. The receiver can verify the integrity of data packets using a data checksum located in the packet header. If a packet is corrupted, TCP asks that it be retransmitted. Strength of TCP is flow control and congestion control which actually lead to slow start. Flow control ensures that the packet sender does not send packets faster than the network can transmit them or the destination host can process them and congestion control ensures that the source sends the number of packets safely by determining the available capacity in the network. TCP also guarantees ordered delivery. It automatically extracts data from incoming packets and inserts it into the byte stream that the networked application reads in the correct order. The application itself does not know that the data arrives divided into packets; rather, it observes only a constant data flow. Thus, the reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets.

A negative aspect of TCP's reliability is slowness. Packets may be delayed for a certain amount of time in order to preserve the ordering guarantees or retransmit the lost packets. Furthermore, when packets are lost, a specified amount of threshold time is initially requisite to determine that the packet is lost. A packet asking the sender to resend the lost packet is then sent to the sender, and the sender is required to send the packet

again. This causes breaks and delays. TCP structure by itself is fairly rigid. This protocol is not suitable for distributed real-time applications [9]. If update messages are sent over TCP, they will be collected from the network and displayed in the virtual world by the receiving end in the order in which they have been sent. However, if one update message is lost or delayed and later update messages are received, the virtual environment will be updated with the latest update messages and the lost or delayed update messages will become obsolete. TCP will send all update messages reliably, but some messages will not need to be retransmitted because they will already be obsolete since subsequent messages will already have been received. Additional update messages would be undesirable in this situation since valuable network resources would be used without any gain. Moreover, delayed and lost update messages would not meet the requirements for close coordination between two or more users in closely coupled and highly synchronized tasks. Thus, TCP may be suitable for networked virtual environments with only a small number of hosts and limited requirements for data rate.

2.4.4 UDP

User Datagram Protocol, UDP, is another popular protocol that is a connectionless transport-layer protocol that belongs to the Internet protocol family. UDP is built on IP and is an interface between IP and upper-layer processes. UDP protocol ports distinguish multiple applications running on a single device. UDP is designed to serve situations in which the properties of TCP cause problems. Unlike TCP, UDP provides no reliability, flow-control, or error-recovery functions for IP. UDP is of a lighter weight than TCP, and UDP headers contain fewer bytes and consume less network overhead than TCP. Obviously, UDP is the simplest protocol overhead. UDP differs from TCP in terms of three main aspects:

1. Unlike TCP, UDP provides transmissions without having direct or peer-to-peer connections between sending and receiving nodes. The sending and receiving nodes do not retain any information or buffer sending or receiving messages concerning the state of the communication session. UDP is useful in situations where the reliability

mechanisms of TCP are not necessary, such as in cases where a higher-layer protocol might provide error and flow control. Whereas TCP uses this information to detect packet loss and dynamically adjusts the rate of data transmission, UDP does not have these properties. The packets are directed from the other host to the address.

2. UDP offers best-effort delivery. UDP does not make any attempt to guarantee reliable data delivery or even ordered delivery. Data is transmitted blindly and packets are received as they arrive. If packets are lost, the protocol does not notice. UDP is a very simple protocol that supplies only the port number and the checksum – which distinguishes several applications at the same time using the network on a single IP address and ensures that the packet does not contain bit errors.

3. The UDP packet format includes source and destination ports, a checksum, the UDP header, and data fields. UDP uses packet-based data semantics. It does not create the illusion of a constant data stream. UDP data is sent and received on a packet-by-packet basis. Sent data chunks must not be too large. If the underlying IP protocol fragments the chunks, certain pieces might be lost in transit, causing the data chunks to arrive corrupted.

UDP causes less overhead compared to TCP, and packets are delivered immediately. Several operating systems impose limits on the amount of simultaneous TCP/IP connections. The operating system does not need to retain any connection information for UDP. These features make this protocol more appropriate than TCP for large-scale distributed systems. One problem with UDP is low security. When a socket receives data on an UDP port, it will receive packets send to it by any host, irrespective of the sender's participation in the current application. TCP can do the same; therefore, the application must be able to distinguish robustly between expected and unexpected packets.

In the network architecture sense, UDP is still a point-to-point connection. Sending the same data to multiple hosts requires that it be sent once for each host. This causes network overhead. UDP suits collaborative virtual environments with higher data requirements and a possibly larger scale than TCP. UDP is particularly relevant to this thesis because it plays a very important part in sending update messages to participants.

In tele-haptic CVE, the interaction stream is abstracted into critical messages, such as last update messages, which can be sent reliably, and regular messages, which can be sent by best effort transport. Moreover, in this project we introduce a novel prediction technique that is able to detect and deal with lost update messages for UDP communication based on history (a buffer in the receiving side) in the networked virtual environments. This helps reduce the impact of UDP's unreliability.

2.4.5 IP Broadcasting

A broadcast is an architecture in which a data packet is destined for multiple participants. Broadcasts can occur both at the data link layer – sending a packet to all participants attached to a particular physical network – and the network layer – sending a packet to all participants attached to a particular logical network. Most IP Broadcasting is built on User Datagram Protocol (UDP) broadcasts. It allows a single message to be sent to all hosts listening to a designated port. The messages are broadcast to one port on each computer in the entire network, but the scope of the broadcast can be controlled by generating a bit mask representing the subnet of hosts that should receive it.

IP Broadcasting has the similarity in packet transmitting strengths as UDP and includes simultaneous delivery to multiple hosts. IP broadcasting also has the same limitations as UDP. In addition, its delivery scope is limited to local area networks. IP broadcasting should not be used over the Internet. The Protocol sends each packet to all hosts, resulting in expensive overhead. Every host on the target subnet must receive and process every broadcast packet, irrespective of the host's participation in the current application. UDP packets are encapsulated inside the IP protocol, which is further encapsulated into LAN protocols such as Ethernet, Token Ring, etc. The only information available at these lower protocol layers is the IP address destination of the packet, not the UDP port number. Thus, a machine can determine whether to keep or discard a packet only after the UDP information has been processed by the operating system, thereby causing processing overhead.

Though broadcasting seems to solve many of the problems of collaborative virtual environments since it is fast and only one message per update message is sent, in practice broadcasting turns out to be a nightmare because every host on the subnetwork, even

those not participating in the simulation, receive and process the broadcast messages. As a result, IP broadcasting suits powerful computing stations, very high bandwidth optical networks, or networked virtual environments with high data requirements and time-sensitive delivery needs running on LAN. It should not be used to communicate beyond the local area network.

2.4.6 IP Multicasting

Internet Protocol (IP) multicast reduces traffic on the network by simultaneously delivering a single stream of information to multiple receivers without adding any additional burden on either the source or the receivers. IP multicasting is built on UDP. It distributes packets only to the nodes that have expressed interest in receiving them and have been configured as members of a multicast group in various scattered subnetworks. It does not send duplicate information over any part of the network. This is achieved by constructing distribution paradigms according to the receiver's interests. The node sending a message that needs to be delivered to multiple hosts sends a single copy of the message to the subnet where the receiving hosts are located. At each network cross-section, the message is sent only to lower subnets, which have participating nodes beneath them. The single message is duplicated at the cross sections and copies are sent down the appropriate paths.

Multicast improves the lessons learned in broadcast. In this architecture, only hosts that are registered with the multicast address receive update messages. The most successful implementation of this approach is the DIS and HLA standard, which will be discussed in detail later. In order to support collaborative virtual environments (CVE) where participants often require delivery of update messages to a set of participants in various scattered networks, IP multicasting is conducted using a reliable transport protocol. Here, a copy of the multicast data is sent to registered participants one by one using TCP/IP unicasts and unreliable transport protocol. The use of network-level multicast is possible through UDP/IP. It does not burden hosts that do not wish to receive multicast data. On the other hand, Internet routers do not yet support IP multicasting. IP multicasting built on UDP also lacks reliability. However, multicasting is the best-known base protocol for large-scale networked virtual environment communication. Already

efforts to incorporate reliable multicasting technology in DIS/HLA environments such as Selectively Reliable Transmission Protocol (SRTP) [32] are underway. IP is well suited to both large-scale peer-to-peer and client-server networked virtual environments, particularly those that use the Internet.

2.4.7 Distributed Interactive Simulation (DIS)

The DIS protocol is an IEEE standard (IEEE 1278) [20] that provides architecture for logical communication among entities in a distributed simulation. Although DIS was developed for military simulation purposes, the protocol was later applied to communication between physical interactions of any type of physical entity. It is also adaptable to general use. The main objective of DIS is to create an infrastructure that joins numerous simulators located all over the world to one single virtual environment. All simulation entities – vehicles, ships, planes, soldiers – transmit their state to others by sending data units defined by DIS standard to the connecting network. In DIS, the entity's state information is exchanged via protocol data units (PDUs), which are defined for a large number of interaction types and are not sent continuously. The entity's state, such as its position and posture, as well as its linear and angular velocities and accelerations, is encapsulated by the Protocol Data Unit (PDU). The PDU is able to include the articulated special component of an entity, such as the orientation of moving parts, by using extrapolation algorithms. This method is known as dead reckoning; it forecasts the intermediate moments between state information transmissions. A variety of dead reckoning algorithms permit listening hosts to obtain the computationally efficient projection of an entity's posture.

Collaborative Virtual environments allow distributed participants to collaborate in closely coupled and synchronized tasks that require point-to-point sockets for tight coupling, and real-time response of physics-based models. The DIS protocol enables efficient live interaction between multiple entities in multiple virtual worlds. When combined with multicast transport, DIS provides solutions for many end-to-end communications requirements. DIS was intended for real-time simulation systems and has been used in simulations consisting of thousands of nodes. Despite its numerous advantages, DIS is not sufficiently broad and is not adaptable enough to meet general

virtual environment requirements. However, DIS is in the process of being replaced by a newer networking system, HLA.

2.4.8 High Level Architecture (HLA)

The Department of Defense (DoD) High Level Architecture (HLA) was designed to facilitate interoperability among simulations and to promote the reuse of simulations and their components. HLA is composed of three major components: HLA Rules – A set of ten basic rules that together describe the general principles defining HLA; HLA Interface Specification – A description of the functional interface between simulations (federates) and the HLA Runtime Infrastructure (RTI); and HLA Object Model Template (OMT) - A specification of the common format and structure for documenting HLA objects.

High Level Architecture [17] was proposed as the support of the next generation of modeling and simulation projects. HLA is not a protocol but rather an architecture, the standard (IEEE 1516), deriving for distributed military-purpose simulation, which provides scalability and general federation/federate and object management services. However, HLA has drawbacks in terms of building large-scale general-purpose collaborative virtual environments because of its object-attribute-based ownership management permit joint manipulation in constraint-based solutions. Many research efforts [35] to realize joint manipulation management in CVE and develop a heterogeneous scalable architecture for large collaborative haptics environments over a non-dedicated channel such as an internet connection are ongoing. HLA designates the abstract interface under which different developers may develop their own protocols. The abstract interface makes HLA systems independent of lower-level protocol implementation.

HLA attempts to limit the type of exchange information as little as possible. It is designed to allow the separation of a specific functionality from general-purpose supporting runtime infrastructure. HLA does not rigidly define the structure of connected entities, nor the interactions between them, as DIS does. Instead, it defines only a model that is used to describe the entities. This makes HLA flexible. New units, elements, objects, and properties are simple to add.

The common system terms defined by HLA [17] are:

1. *Federation.* A federation is a combined system or unified simulation environment composed of individual applications or any number of physically distributed simulation systems, usually simulators and visualization processes that are connected together. For instance, a federation can be weather and army combat simulation systems brought together over the network into a unified simulation environment to address the needs of new war simulation.
2. *Federate.* A federate is one of the networked simulation entities brought to form the federation. For instance, a federate can be the simulation of a tank, ship, or airplane, or it can include the simulation of a whole weather system, or it can be a visualization process. A federate can be thought of as either a process or an individual complete simulation joined to the federation.
3. *Federation execution.* The HLA Federation Development and Execution Process (FEDEP) Model describes a generalized process for building HLA federations by providing lower-level development practices native to each individual application or simulation. These can be easily integrated. Federation execution is simply the session of a federation of federates executing together.

Runtime Infrastructure (RTI) [6,17] , which implements High Level Architecture (HLA), is very likely to allow distributed participants to collaborate in closely coupled and highly synchronized tasks [35]. Therefore, HLA/RTI can be used as middle-ware to exchange haptic information among federates (participants in a group interact with the same virtual object or share the same virtual environments). Multicasting is a technique used to provide scalability, runtime performance, minimized delay, and reliability in HLA/RTI supported collaboration virtual environments.

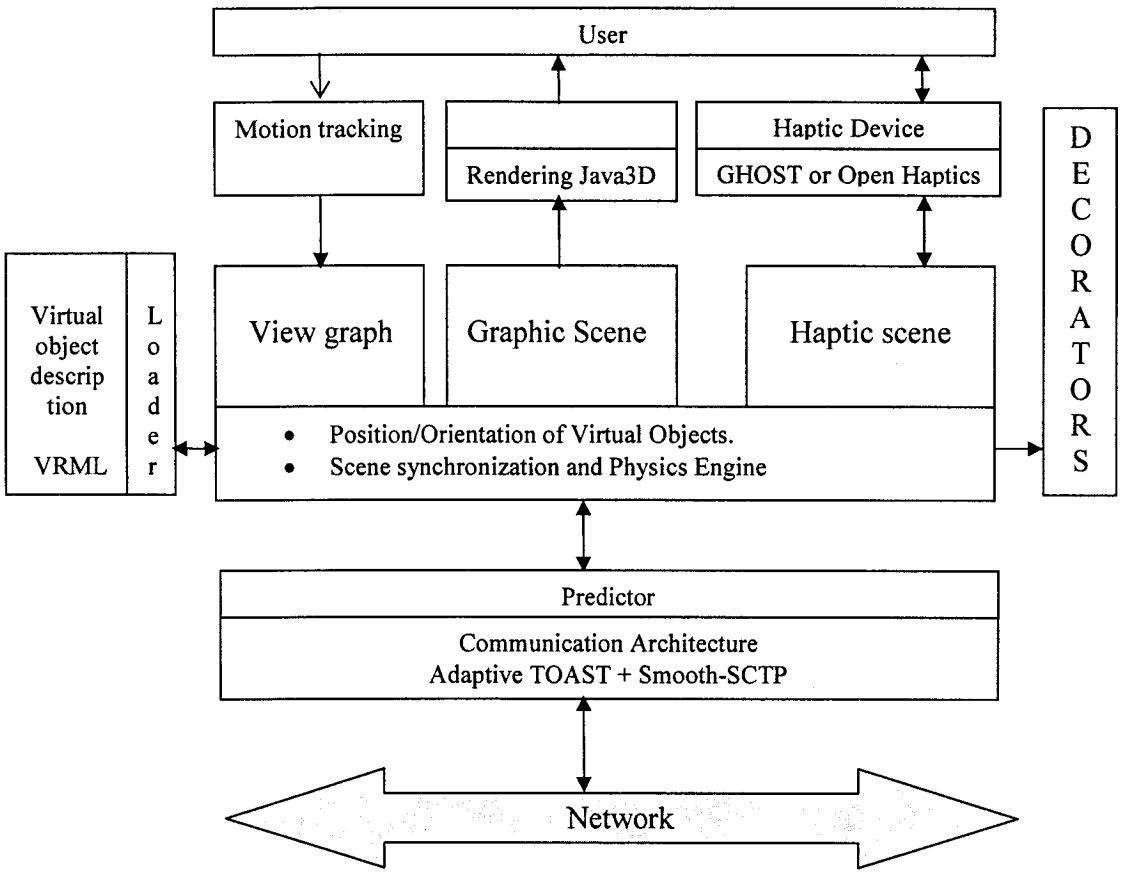


Figure 2.5: Haptic Incorporated virtual environment Application Architecture.

2.5 The Architecture of a Haptic Incorporated Collaborative Virtual Environment

Haptics control and rendering generally require a high sampling rate and a low latency. The aim of this thesis is to develop a scalable architecture over a computer network for collaborative haptics environments. Tele-haptic platforms rely on hard real-time operating systems in order to guarantee the stability of the control loop and minimize delays in network stack. The haptic feedback in CVE is the result of a combination of different architectures. A participant in such a CVE with a haptic device is comprised of three parts: the haptic device, stream interaction transmission protocol over a network,

and virtual environment graphic systems. To accomplish the collaborative virtual environments using haptic devices, the applications require:

- *Shared virtual environments*: These are a description of the 3D virtual world and the objects in a machine-readable format. Basic descriptions include the shape of objects, their colors, lighting properties, positions, and behaviors. A loader reads the machine-readable format and represents the virtual objects in the virtual universe. In graphics-rendering, a participant is able to choose either a VRML-enabled web browser or a Java3D-based browser.

- *Consistency of virtual environment rendering*: This plays a vital role in the creation of a CVE to control virtual scenes and communicate with other virtual environments through middle ware.

- *A graphic rendering program*: This displays the virtual environment at a frame rate that gives the user the illusion that either he is moving or the virtual environment is dynamic without intersection. It must provide primitives for manipulation and representation of geometric objects in a three-dimensional universe. In this thesis, both Java3d and VRML are used to display three-dimensional virtual environments and interactive virtual objects.

- *An interface with haptic devices*: The PHANToM haptic interface uses position information input by the participant to determine what forces to relay to the participants via its three motors. SensAble Technologies Inc. provides a General Haptics Open Software Toolkit (GHOST® SDK) and The OpenHaptics toolkit API to represent the haptic environment as a hierarchical collection of geometric objects and spatial effects. Both interfaces use OpenGL and 3D graphics and require a much higher update rate (1000 Hz) than the graphical rendering (10 to 100 Hz). This is the reason why two separate scenes are usually created (a haptic scene and a graphic scene) and are updated at different rates. The application must maintain synchronization

between both scenes. In addition, the 1000 Hz servo loop performs the following enumerated functions:

1. Updates the PHANToM participant position in virtual environments;
2. Updates the dynamic state of all dynamic virtual objects;
3. Detects collisions in the virtual environment; and
4. Relays the resultant force feedback to the PHANToM.

- *An interface with other virtual environments*: This is a single object among multiple geographically distant participants or other participants sharing the same virtual environment. It represents a significant research challenge and inevitably requires significant usage of both network and computer application-based facilities. Computer network resources are used to control a single shared object, which is in fact the virtual object passed back and forth between individual participants. Resources are also used to represent the consistency of the dynamicity of the virtual reality and manage the interaction between the objects (collision detection) and the physical laws (physics engine). Figure 2.5 shows the architecture of haptic-incorporated collaborative virtual environments and other components interoperate.

- *The predictor*: A prediction technique is used in conjunction with both a decorator and a networking-level solution to moderate the effect of simulation lag on a Haptic-based CVE environment where two users must finish a given task collaboratively. The main contribution of this thesis is in the design of the predictive algorithm, which takes into account the importance of an update message from a synchronization perspective and couples it with the decorator.

- *Decorators in collaborative haptic applications*: These are visual cues, preferably embedded non-intrusively within the user interface, that reveal limitations in the collaborative session. In our scheme, while buffering the update messages, the predictor collects timestamps, indexes, stream IDs,

sequence numbers, and the states of the objects of the update messages, and invokes the following possible decorators: the Jitter decorator, the Direction decorator, and the Trajectory decorator.

2.6 Graphic Scene

Creating a well-performing real-time visualization component for the virtual environment system is a fundamental requirement of every virtual environment application. In this section, the basics of a real-time three-dimensional graphic system, including the construction of three-dimensional graphics, will be demonstrated. In addition, we present the technological solutions used in this thesis to meet this requirement.

2.6.1 VRML

The graphic scene of a virtual environment must be described and requires a rendering mechanism in order to be displayed. Descriptions of virtual objects can be either hard-coded directly into the rendering architecture (this is usually acceptable for simple shapes) or stored in an external file. One format used for these files is Virtual Reality Modeling Language (VRML) [43] which is able to describe three-dimensional image sequences and possible user interaction. This format uses markup syntax to describe basic shapes (cube, cone, sphere, and cylinder) and a set of points and their properties (color, texture, position, and lighting). With the help of VRML, we can build a sequence of visual images into web settings with an apparent three-dimension scene with which a user can interact by viewing, rotating, and other interaction.

Objects can be embedded within one another in a hierarchy, thereby forming logical clusters of objects that move together. The scene graph is composed of nodes or objects. These nodes or objects may have routes between them, according to the scene graph hierarchy, which define the interactions or collisions of one object with another. Objects have fields that define the actual value associated with them. Within

the scene graph, certain special objects able to allow fields to be pushed and popped to control certain object parameters.

VRML able to allow a three-dimension world to be delivered over the World Wide Web and browsers can interpret the standard text files of VRML as browsers do the HTML text files. Like HTML, VRML has been designed to be platform independent and to work over low bandwidth connections. While HTML files describe a 2D page containing various text constructs, VRML files describe a 3D space, or "world". A browser that may be equipped with a plug-in for a web browser or a helper application where users can explore a 3D world, zooming in and out, moving around and interacting with the virtual environment, can parse VRML files. VRML does not need a very high bandwidth to transmit three-dimension graphics; even it allows complex three-dimension graphics to be transmitted over Internet without a very high bandwidth. Recent additions to the VRML specifications will users to make virtual worlds that will be available over a shared computer network, such as the Internet and to interact with virtual objects in the virtual environment. Therefore, as computer language, VRML may likely to be the key ingredients that will describe dynamic and interactive virtual objects, of a collaborative virtual environment.

2.6.2 OpenGL

OpenGL is the only low-level rendering and modeling software graphics library specification with wide hardware support available on all major and diverse exotic platforms. Like several other graphical libraries and toolkits, OpenGL was originally developed by Silicon Graphics, Inc in 1992. It was designed as a multi-purpose, platform-independent graphics API, an Application Programming Interface [36]. OpenGL is the only basic and widely accepted 3D graphics API that programs other than Windows, such as Linux, Unix, and Irix platforms, support and allow for use with the C and C++ programming language, in addition, there are also bindings several other programming languages such as Java, Tcl, Ada, and FORTRAN.

OpenGL is a low-level API. It supports polygons and lighting in the form of Gouraud shading, textures, z-buffering, transparency, and other basic 3D elements –

points, lines, images and bitmaps. OpenGL provides a set of commands that allows the specification of geometric objects in two or three dimensions. Using the primitives, together with commands, it controls how these objects are rendered in the frame buffer. However, it does not include any loaders for even the most common image files, or geometry formats. 3D objects must be presented as polygons in OpenGL. OpenGL does not include any interface to window systems on any platform because the 3D standard OpenGL development and usage libraries are free.

OpenGL APIs able to construct a portable, dynamic and interactive three-dimensional environments and virtual objects but these APIs are very large and complex. Sun's Java3D web site [39] claims that API enables ordinary programmers to write three-dimensional graphics, but the size and complexity of the API belies that assertion. Although, low-level OpenGL API based language support for graphics, audio, and networking abstractions, the cost of development limits the rate of experimentation and restricts the developers to the application in collaborative virtual environments.

2.6.3 DirectX

DirectX is Microsoft's combination of multimedia application programming interfaces for Windows operating systems. DirectX is a set of APIs for developing powerful full-colour graphics, video, sound, three-dimension animation, and network play application. DirectX provides a consistent interface to devices across different hardware platforms by abstracting away the underlying hardware by using the Hardware Abstraction Layer (HAL) and Hardware Emulation Layer (HEL). Both HAL and HEL allow the DirectX user use the hardware functionality that might not even be available on the system. DirectX also has fast and low level libraries, which access multimedia hardware in a device independent manner.

The APIs of DirectX support the functionality DirectPlay that provides networking and communication services in a transport independent manner. DirectPlay offers a consistent interface for the developer irrespective of whether the transport mechanism is TCP/IP, IPX or modem. DirectPlay and the HLA RTI, both have similarities in providing an abstract layer to the network, executing federation

with each federate interacting with one another through a communication medium, and providing interfaces and methods to mange sessions/federations. Since DirectPlay interfaces in distributed application are analogues to the application of HLA/RTI, with good application design, DirectPlay merges with other DirectX components most likely to create an exciting Haptic Collaborative Virtual Environments.

Microsoft is a profit-seeking organization, and its foothold on users is strengthened by its quick response to the needs of the 3D programming society and to capture the huge gaming market. For these reasons, DirectX has evolved much faster than OpenGL. It currently supports more sophisticated effects for real-time 3D rendering than OpenGL. DirectX also includes APIs for sound programming and basic input devices. The disadvantage of Direct X is the same property that helped it evolve quickly: its complete dependence on the Windows operating system. No other platforms support it. Nowadays, DirectX is heavily used in the gaming industry. The newest version of DirectX can be obtained at no cost from Microsoft's www-pages.

2.6.4 Java3D

Java3D is part of the Java Media APIs introduced by Sun Microsystems Inc. to allow developers to write a single piece of code, based on Java Media APIs, that is designed to run in various platforms. The Java3D API is a set of classes for writing three-dimensional interactive graphics applications and Internet-based applets which help to create and manipulate three-dimensional geometry. There are two Java3D variants: one is implemented on top of OpenGL, the other above DirectX Graphics. The low-level API handles native rendering at the vertex and pixel levels, while three-dimension scene, application logic, and scene interaction are carried out by Java code. The above dual approach encourages application portability, hardware independence, and high-speed rendering. Since a Java3D program runs at the same level as any other Java program/applet in the virtual machine, the Java3D-based program becomes consistent and controlled through calling the Java 3D API form any Java program. Java3D provides high-level constructs for creating and manipulating 3D geometry and building the structures used in graphics sound rendering systems in order to define and render a very large virtual world.

The implementation of the rendering part of this project has been done by using the powerful classes of Java3D because of this program's compatibility and easy interfacing with Java. Moreover, java classes have been extensively used in the transport layer protocols and comparison framework of our CVE application. Both Java and Java3D were developed and are maintained by SUN Microsystems; they provide a well-documented API with numerous tools for such implementation [38]. In addition, Java3D provides functions such as distance calculation and collision detection. The position values of the virtual object are periodically monitored and sent to distributed users via the computer network. If in the collaborative virtual environment the object position approaches an obstacle, the geometry of this obstacle is sent to participants, but the actual collision detection can be performed by the haptics with high refresh rates, which would be complicated in networked virtual environments. Only the geometry of obstacles that are close to the object's position are sent to the distributed participants; collision calculation will be very straightforward when using suitable prediction for the future state of the object.

Java3D's scene graph is a directed acyclic graph and has a parent-child relationship between most of its nodes without any loops. It is possible for nodes to be shared in a graph in order to create duplicate geometry details. The scene graph for the box-carrying application is shown in Figure 2.6. The box-carrying application contains nodes for a ground, a box (cube), and two hooks (cylinder). Each node utilizes shape and colour node components, and hook shape information is shared.

In order to create the virtual reality systems in this project, Java3D instantiates a scene graph that includes VirtualUniverse and Locale objects and a complete view branch graph. The VirtualUniverse and Locale Objects contain a viewer object. The view branch graph includes Canvas and can be thought of as the conceptual rectangle image plate, which renders the 3D world into a two-dimensional frame of computer display. Canvas can be added to a Java graphic interface such as an instance from Javx.swing.JFrame.

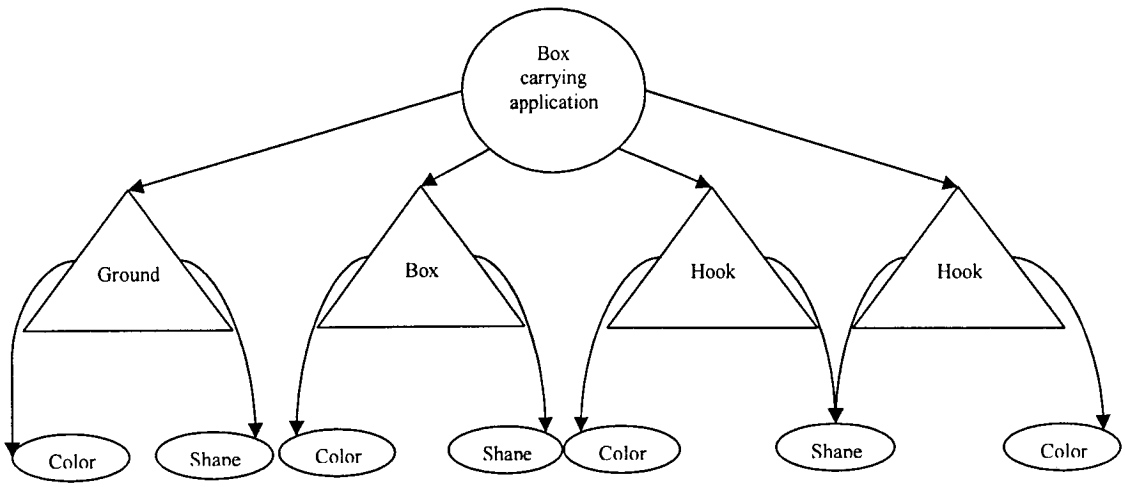


Figure 2.6: Scene Graph for Box carrying application.

The essential elements for creating a Java3d scene that add virtual objects to the universe are described:

- *The universe*: This object allows us to link the instance of Canvas3D with the scene to display. It contains a locale, a viewing platform that defines viewing angle, clipping planes, viewer position, and a viewer for rendering. These are among other settings that are already defined in SimpleUniverse instances.
- *A locale*: The VirtualUniverse objects has a list of Locale objects, which provide a set of high-resolution coordinates in the virtual universe. Using the *Locale*, it is possible to determine the location of visual objects in the universe.
- *Canvas3D*: This AWT component derives from the class java.awt.Canvas and defines a region of the screen for the visualization of a 3D scene. The Canvas3D node is a Java GUI component that allows the 2D image to be placed inside a Java application or applet.
- *Rendering a 3D scene*: The virtual universe consists of various different shapes which can be simple geometries such as a sphere or cylinder, or more complex structures. Transformations such as translation, rotation, and scaling applied to the different shapes allow the addition of objects in the virtual universe or the animation of interactions with users. Shape3D contains geometric data as well as appearance definitions of the object. A Scene Graph

tree, which holds information concerning the scene, represents both the shapes and transformations of a virtual object. Figure 2.8 shows the scene graph of a virtual object.

- *Branch group*: The root of the Scene Graph tree is an instance of BranchGroup. Java3D shares the concept of a SceneGraph with other types of 3D programming languages and file format, like VRML. BranchGroup allows children to be added to or removed from the graph at any time, while TransformGroup permits the position and orientation of its children to be changed. For example, a virtual object such as a box could be described in a branch group which may be added to or removed from other branch groups, and added to or removed from the locale during runtime.
- *Transform group*: Transform group is a subclass of Group class and has a collection of child node objects. A transform group is added as a child either to another transform group or a branch group. TransformGroup objects are used to group several objects that hold geometric transformations, such as translation and rotation. A generalized transform object (point or vector) is represented internally as a 4x4 double precision floating-point matrix. The mathematical representation is row major, and a Transform3D object is used to specify the transformation of a TransformGroup object. The 4x4 matrix (the transformation matrix) can be defined as

$$\begin{pmatrix} T_{00} & T_{01} & T_{02} & T_{03} \\ T_{10} & T_{11} & T_{12} & T_{13} \\ T_{20} & T_{21} & T_{22} & T_{23} \\ T_{30} & T_{31} & T_{32} & T_{33} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix}$$

with the resulting co-ordinates

$$\begin{pmatrix} x' = T_{00}.x + T_{01}.y + T_{02}.z + T_{03}.w \\ y' = T_{10}.x + T_{11}.y + T_{12}.z + T_{13}.w \\ z' = T_{20}.x + T_{21}.y + T_{22}.z + T_{23}.w \\ w' = T_{30}.x + T_{31}.y + T_{32}.z + T_{33}.w \end{pmatrix}$$

Therefore, in a three-dimensional coordinate representation, a point or vector is translated from position (x,y,z) to position (x′ , y′ , z′) with the transformation matrix operation as Figure 2.7. In the case of vectors, w is set to 0 and only the 9 coefficients in the upper left corner of the matrix are relevant.

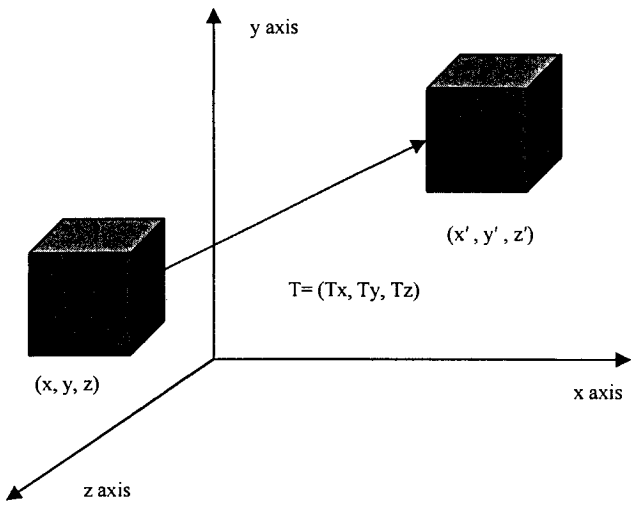


Figure 2.7: Translating an object with translation vector **T**.

For example, the object is translated and the transformation matrix is equivalent to the three equations

$$x' = x + Tx, \quad y' = y + Ty, \quad z' = z + Tz$$

To specify a rotation transformation for an object, the object must designate an axis of rotation (around which the object is rotated) and the amount of angular rotation. The three-dimensional z-axis rotation equations are expressed as:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1] \begin{pmatrix} \text{Cos}\theta & \text{sin}\theta & 0 & 0 \\ \text{-Sin}\theta & \text{cos}\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Parameter θ specifies the rotation angle.

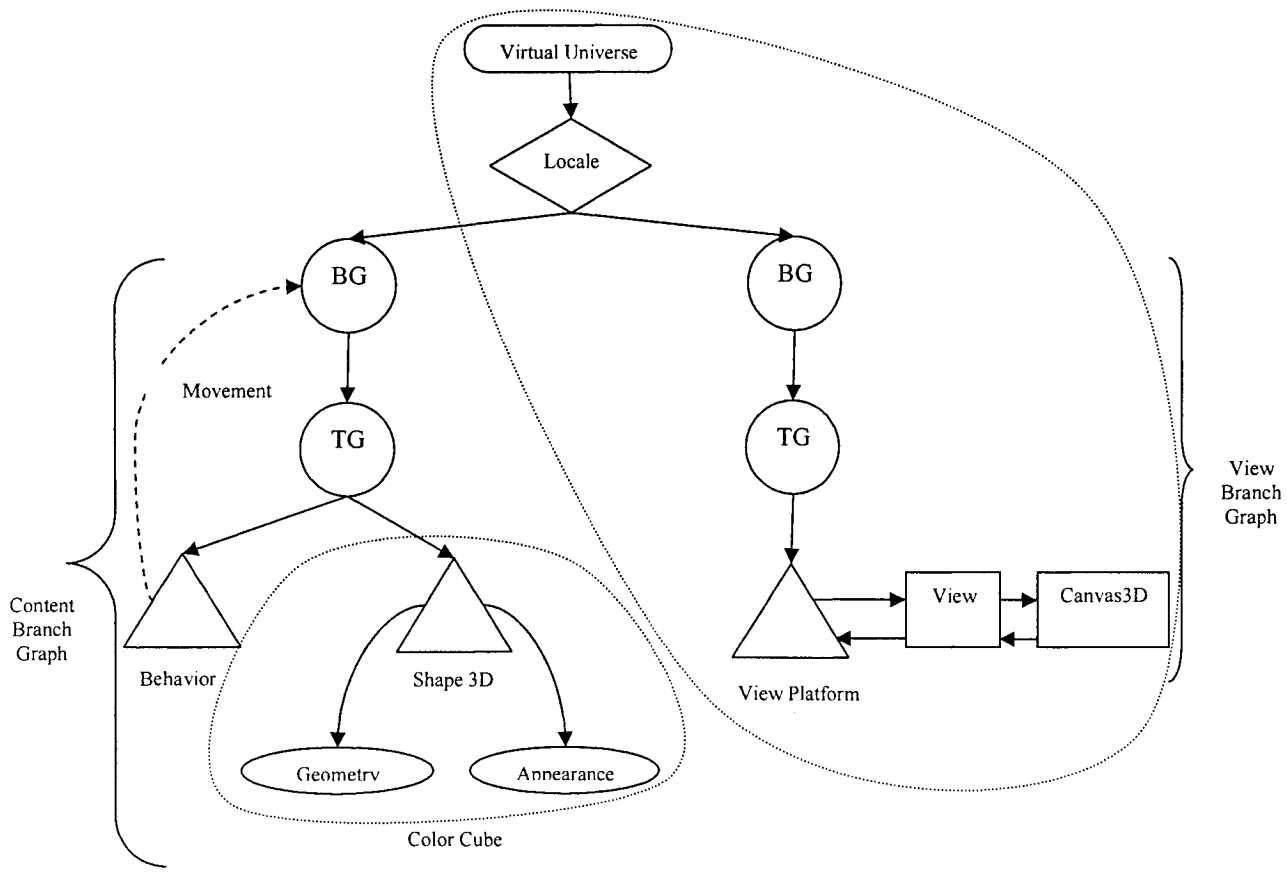


Figure 2.8: Scene Graph for Box carrying application.

Furthermore, a virtual object is translated and rotated, or is rotated around two axes. In either case, two different transformations are specified for a single visual object. The two transformations can be combined into one transformation matrix and held by a single TransformGroup. An example defined here is the rotation of a cube around both the x and y axis. Two Transform3D objects – one for each rotation – are combined by multiplying the Transform3D objects. TransformGroup object loads the combined transformation. Unlike branch groups, transform groups cannot be added or removed freely. Once the BranchGroup loads the TransformGroup , no addition or deletion is possible.

- *Primitive geometries, shape node, and environment nodes*: Primitive geometries define common graphical objects, such as boxes, cones, cylinders and spheres, by specifying their geometrical properties. A shape node employs Shape3D (or its subclasses) as a container for geometry (usually specified as a set of points, lines, triangles, or squares), and Appearance node components. There are many Appearance subclasses, including ones for colour, texture, and transparency. Environmental nodes handle environmental concerns such as lighting, sound, and behaviors that change the virtual environment. Shape node, behavior nodes, and primitive geometries are added as the children of a transform group, and are modified according to the transformations of their parent. In the box-carrying application, the box (the Color Cube) is composed of a Shape3D node and associated geometry and appearance components. Its transformation is carried out by a Behavior node, which affects the TransformGroup parent of the ColorCube's shape.

The Figure 2.8 mentions the partial branch graph of the first application. The right-hand branch below Locale is the view branch graph and specifies the participant's position, orientation, and perspective as he or she looks into the virtual world from the physical world such as the computer monitor.

2.7 Haptic Devices

Haptic devices facilitate interaction with a three-dimensional virtual object by incorporating the sense of touch into the computer interface. They provide users with more realistic and real-time access to people, environments, and information. When haptic devices are used over a network to manipulate the virtual environment in a natural and effective way, they enhance the sensation of "touch" and provide information such as stiffness and texture of objects. Such remote collaborations over a network are termed tele-haptics, and highlight collaboration problems. To address collaboration problems, learn more about haptic devices, and explore their uses in tele-haptics collaboration, this thesis also considers the issues of haptics properties. This section presents haptic devices; few of them have been used in this project. The haptic rendering mechanism uses software to interface with the user so that he or she can 'feel' and represent the force-rendering.

2.7.1 PHANToM

PHANToM (Personal Haptic Interface Mechanism) [27] is one of the most widely-used haptic devices that allow the user to feel forces of interaction. It helps users collaborate by transmitting a sense of touch, which demonstrates a successful collaboration in a complex disassembly task simulation [3]. Two more successful applications are removing the fuel tank of an aircraft with screwdrivers and wrenches, and identifying friends and foes on a monitor using a gear-shift like a haptic device. This device is a pen-shaped device like a small desk lamp and has a stylus grip or fingertip thimble, which provides force feedback between virtual objects and the user.

The Phantom Haptic interface allows motion along six degrees of freedom as an input, and three degrees of freedom as an output, with a maximum exertable force of approximately 8.5 Newtons. It does this via three motors, which control the x, y, and z forces exerted on the tip of the user's finger. The turning force (torque) of the motors is transmitted through a cable transmission to a stiff, lightweight linkage. At the end of this linkage is a passive, three degrees of freedom gimbal attached to a

thimble. The passive gimbal allows the thimble to rotate, and creates the sensation of any comfortable orientation. Therefore, the six numbers of independent variables can describe the phantom's movement, which is needed to describe the state of a physical object. We can consider one example: consider a solid cube, too lightweight to be easily lifted or rotated, that remains on the desk. The object (cube) can be moved in any direction or rotated around any axis. Therefore, it has three DOF in translation as well as three DOF in rotation. In total it has six DOF: exactly six variables are needed to describe its position. On the other hand, a solid object is too heavy to be moved. The object can be moved in any direction but not rotated around any axis. Therefore, the object has three DOF in translation and zero DOF in rotation. In total, it has three DOF: three variables are needed to describe its position. The PHANToM can sense the movements of the user hands, as well as the rotations the hand makes, in all directions of space. It has, therefore, 6 DOF in input (three for the translation, three for the rotation). It therefore has low inertia, low friction, and no unbalanced weight, and can allow movement anywhere within its workspace. These phenomena help users feel unimpeded and undistracted in the presence of the device, and also allow free motion of the hand.

On the other hand, in the case of output (the force feedback part), only the translation components may be exerted. It is possible to constrain the position of the pen-part of the device in all directions of space, but rotations cannot be constrained. It is possible to make the user feel force, but not torque. As this haptic interface cannot supply twisting forces to the fingertip, it is modeled as a point or frictionless sphere within the virtual universe. It should be noted that other haptic devices provide six DOF both in input and output, but they are considerably bulkier and more expensive.

Solid objects can be haptically rendered by constructing a plane tangential to the point on the surface where the object is touched. The application of force feedback ensures that the user cannot penetrate this boundary. Computer-generated virtual objects can be 'touched' by the user, and it is very important to correctly design the friction and texture models, which is pertinent to the surfaces of the objects in the virtual environment. Then the touch can convey the real feeling of the objects.

Without these surface properties, all of the objects will feel slippery as though they were made from ice. Friction models for objects generate forces tangential to the surface and in the direction opposite to that in which the haptic probe is moving. Several kinds of friction models, such as Coulomb friction (static and kinetic), viscous friction (friction proportional to velocity), and drag friction (friction proportional to squared velocity), exist.

A realistic texture can be mapped to a virtual surface by means of changing the value of the surface normal vector. Doing so creates the haptic illusion of feeling a textured surface. A deterministic or stochastic function, or a combination of these two functions, allows the texture force vector to be computed. The choice of friction model is crucial for successful tele-colaboration, but this is not the topic at hand; instead, friction model choice should be considered as future work.

PHANTOM® Omni™

The SensAble Technologies PHANTOM® product line of haptic devices makes it possible for users to touch and manipulate virtual objects. The PHANTOM® Omni™ model is the most cost-effective haptic device currently available. Portable design, compact footprint, and IEEE-1394 FirWire® port interface ensure quick installation and ease of use. It is compatible with 3D Touch™. The Omni is an easy choice because it is inexpensive and has good force-feedback qualities, and is also an easy-to-use programming API.

The OpenHaptics toolkit is used to control the PHANToM Omni haptic device. The forces are small and the workspace for the haptic device is enough to realistically mimic a real situation. The only limitation with using this device is the limited degrees of freedom (DOF) in a force model. The Omni delivers six DOF position information such as x, y, z, pitch, roll, and yaw from sensors, but can only control the actuators in three DOF such as x, y, z, which leads to the limitation that only forces, not torques, can be sent back to the user.

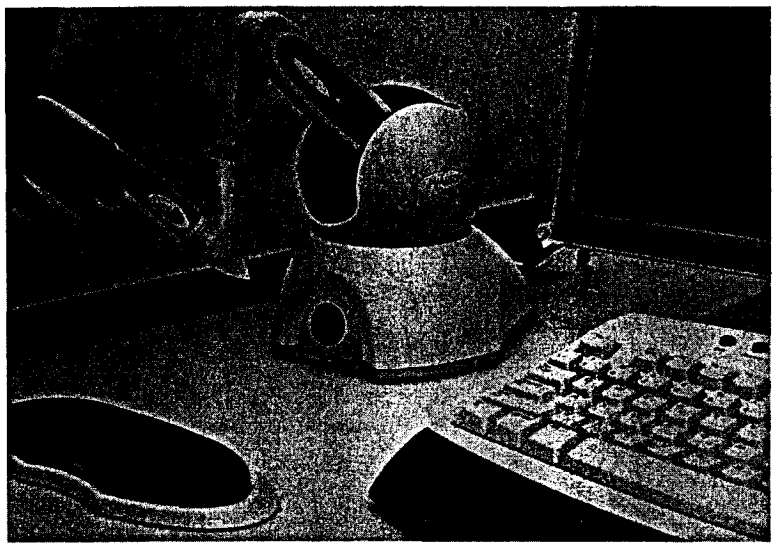


Figure 2.9: The PHANTOM® Omni™ model.

PHANToM Desktop™ Haptic Device

The award-wining PHANToM® Desktop™ haptic device provides an affordable desktop solution and is ideal for performing certain types of haptic research. The PHANToM Desktop provides precision positioning input and high fidelity force-feedback output. Portable design, a compact footprint, and simple parallel port interface ensure quick installation and ease-of-use.

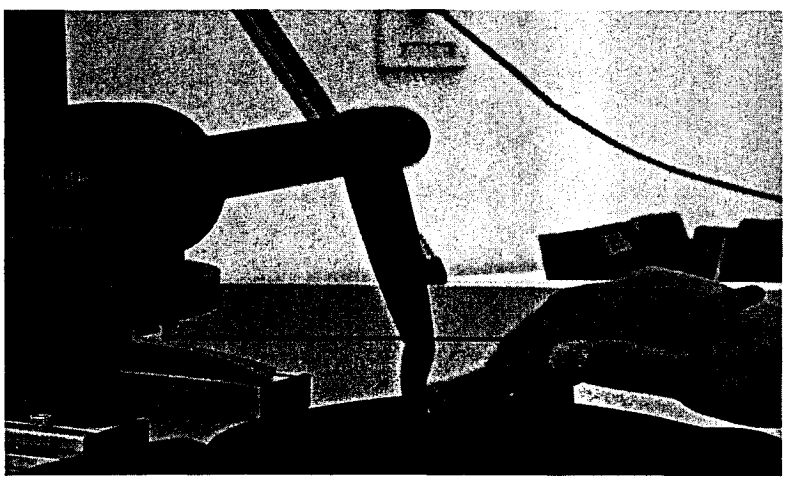


Figure 2.10: PHANTOM Desktop™ Haptic Device.

The OpenHaptics toolkit or GHOST® SDK is used to control the Desktop™ Haptic Device. The forces are high-fidelity and the workspace for the haptic device is

enough to realistically mimic the real world. Like Omni, the only limitation with using this device is the limited degrees of freedom (DOF) in a force model. The Desktop™ Haptic Device provides six DOF position information such as x, y, z, pitch, roll, and yaw from sensors, but can only control the actuators in three DOF such as x, y, z, which leads to the limitation that only forces, not torques, can be sent back to the user.

PHANToM® Premium Model Haptic Devices

The PHANToM Premium haptic devices provide a broad range of force feedback workspace, range of motion, stiffness, and motor force in order to accommodate the specific requirements of different research projects. The Premium models 1.0, 1.5, and 3.0 provide three degrees of freedom positional sensing and three degrees of freedom force feedback. An encoder stylus, which may be purchased separately for the 1.0, 1.5, and the 3.0 devices, enables the measurement of an additional three degrees of positional sensing (pitch, roll, and yaw). A PHANToM Premium 1.5 HF (High Force) device is available, and includes a built-in gearbox option that provides higher forces.

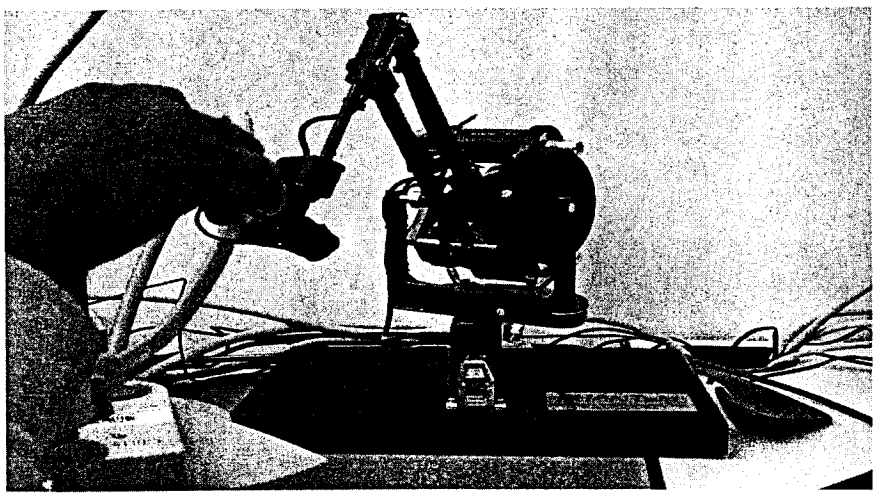


Figure 2.11: The PHANToM Premium haptic devices.

The PHANToM Premium 1.5/6DOF, 1.5 High Force/6DOF, and 3.0/6DOF devices allow users to explore application areas that require force feedback in six degrees of freedom (6DOF). Examples include virtual assembly, virtual prototyping, maintenance path planning, teleoperation, and molecular modeling. All Premium models connect to the PC via the parallel port (EPP) interface. It supports Windows® XP, Linux® platforms.

2.7.2 Architecture of a Haptic Device Interface

In this project, the architecture of a Haptic Device Interface includes Haptic rendering and force modeling. The architecture of a haptic rendering system is an important issue in the collaborative session with regard to communication delays and the accuracy and stability of haptic rendering. The haptic rendering loop should be operated at a 1000Hz refresh rate in order to be able to give credible sensations of the simulated object in virtual reality. If the update frequency is less than 1000 Hz, the surface of the virtual object becomes unstable or soft. This substantially reduces the level of realism in virtual environments. The primary requirements for haptic rendering in virtual reality are shown in Figure 2.12.

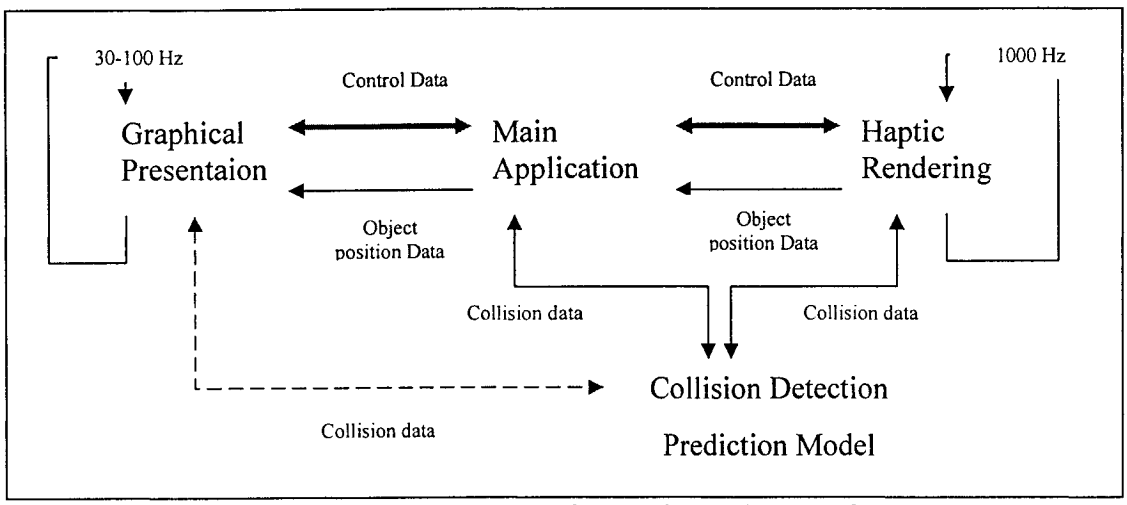


Figure 2.12: Architecture of a Haptic Device Interface

The main application provides separate threads to execute processes concurrently. At first, the main application initializes and controls the haptic rendering systems, then it offers

different threads to haptic and graphical environments in order to maintain synchronization among them. Using a separate thread, the main application sends position/orientation data to the graphical presentation process at a rate 25-100 Hz, and to the haptic rendering process at a rate 1000 Hz, in order to match haptic and graphical environments. Any inconsistency or delay between haptic and visual environments creates instability and reduces the level of sensitivity.

In the box-carrying and tracheostomy surgery application, detecting collisions between the haptic interface device and the virtual object (rendered surface) is a very important task as it allows for a reduction in computation time and gives a realistic response to the user. The calculation of the movement of the virtual box, knife, cotton, or scalpel with which the PHANTOM is in contact requires both a collision detection system to determine the contact surface on the virtual object and a kinematic model for movement of the virtual objects based on the PHANTOM's motion. In order to develop an algorithm for collision detection, there are three recognized approaches, including point, line segment, or the 3D object [2] shape of the probe. For the box-carrying application and tracheostomy, the probe is best as a point-based interaction model; only a force is sent back to the 3DOF haptic device.

The task of collision detection is to determine when the point is in contact with any part of the surface of the virtual object – whether the point is passing through or touching the surface. In this thesis, collision detection techniques include the calculation of the distance between the current PHANTOM position and the virtual object's surface during iteration of the servo loop and force feedback, which is sent to the PHANTOM. Consider three different approaches that deal with Collision and the probable solution. Collision detection may be a separate process handled by the main application, a part of the haptic rendering that requires a very high sampling rate, or part of the graphic rendering. Collision detection may be a challenging task when haptic rendering is applied to a networked virtual environment; a very high update rate, or a complex 3D visual model, may be needed. In such a scenario, a prediction model associated with the main application can execute a collision detection process to meet different level of collision response for different applications. In particular, solid, elastic, or deformable objects have specific collision properties. The other solution, which is based on graphics rendering,

deals with Java3D functions in distance calculation and collision detection. The position values of the virtual object are periodically monitored and sent to distributed users via the computer network. If in the collaborative virtual environment the object position approaches an obstacle, the geometry of that obstacle is sent to participants, but the actual collision detection can be performed by the haptics with high refresh rates, which would be complicated in networked virtual environments. Only the geometry of obstacles that are close to the object's position/orientation is sent to the distributed participants; the collision calculation will be very straightforward when using suitable prediction of the future state of the object. The prediction-based collision detection algorithm is our future research topic, and we discussed it to clarify the needs for calculating collision detection in successful tele-haptic class of applications.

The software architecture definition for haptic environments is an important issue to render haptic prototypes with complex shapes. Haptic rendering requires a huge computing power that includes both hardware and software. The ability of haptic rendering depends on the architecture, which deals with the following primary software components:

- 3D graphics renderer
- haptic renderer
- collision detection technique

A 3D graphics renderer and its important role in virtual prototyping are described earlier. A haptic renderer provides the position/orientation of virtual objects, the force and tactile feedback generated for the user through use of the haptic device, and the kind of haptic properties that can be simulated. Software architecture design requires basic components for at least haptic and visual rendering, and proper allocation of software components between distributed computers. Two haptic rendering software architectures are provided below.

General Haptic Open Software Toolkit (GHOST®)

The General Haptic Open Software Toolkit (GHOST®) [32] API has been developed by the haptic manufacturer of the PHANTOM. It is an object-oriented three-dimensional

haptic toolkit used with a PHANTOM haptic device that offers a C++ library of object-oriented objects and methods. It can be used to represent the haptic environment as an hierarchical collection of geometric objects, spatial effects, collision detection, force/tactile feedback, and interface with the hardware. GHOST SDK help developers specify object geometry and properties, or global haptic effects, by using a haptic scene graph, which is a hierarchical collection of nodes. Moreover, the GHOST SDK is highly portable in different haptic interaction devices such as PHANTOM® Premium, PHANTOM® Desktop™, etc. The main objective of GHOST in prototyping the haptic rendering system is given below:

- Shape primitives: spheres, cones, cubes, and cylinders
- Complex shapes constructed from separate planes
- Collision detection algorithm
- Buttons, dials, sliders, and consistency in shape, which make simplicity in creating haptic rendering, especially in complex prototypes
- Special effects: inertia, buzz, and spring constraint

In CVE, instead of generating visual representations of objects within the haptic scene graph, GHOST SDK provides callback mechanism to facilitate integration between the haptic and graphic domains. In my implementation, the GHOST SDK has been successfully used with Java 3D for rendering visual objects. Instead of presenting an extensive description of the GHOST API, it enhances the few features used in implementation, along with the code that has been used.

Classes gstScene, gstSeparator and gstPHANToM

The API allows the programmer to build a haptic virtual world with the same node mechanism as the one described before, the role of the nodes being played by instances of gstScene, gst.Separator, and gstPHANToM. gstScene manages the haptic scene graph and simulation. gstSeparator is an hierarchical node class that provides grouping of nodes into a subclass in a haptic environment, and *gstPHANToM* provides the PHANToM device interface in scene graph.

```
scene = new gstScene;
root = new gstSeparator;
phantom = new gstPHANToM ("Default PHANToM");
root->addChild (phantom);
scene->setRoot (root);
isInGrapArea=false;
scene->startServoLoop ( );
```

Other separators and objects belonging to the haptic scene will be added as children of the root as gstSeparator, which is used to add, query, or remove children by the methods gstSeparator::addChild, gstSeparator::getChild, or gstSeparator::removeChild. The public member startServoLoop() begins a new thread of execution that loops at a frequency of 1kHz. Every millisecond, the position of dynamic objects present in the scene is updated, callbacks are generated for the objects whose position or orientation has changed, and the force that must be felt by the user is computed and applied to the PHANTOM. The executing procedure continues to run until it is explicitly stopped using the gstScene::stopServoLoop method, an error occurs in the servo loop, or the application exits.

Finally, gstPHANToM must be instantiated and added as a child to the scene root. It is impossible to directly specify forces that have to be applied to the hardware. Those forces are computed and applied by GHOST according to the specified force fields and to reaction forces that are encountered when the PHANToM reaches a surface belonging to the haptic scene.

```
gstPoint phantomPos = phantom->getPosition_WC ( );
gstVector phantomAngles = phantom->getGimbalAngles ( );
gstVector reactionForce = phantom->getReactionForce_WC ( );
jdouble *resultArray = (env)->GetDoubleArrayElements (finalresult, 0);
```

gstPHANToM contains methods that return the device's current position and orientation, update the dynamic state of all gstDynamic objects, detect collisions of gstPHANToM nodes with geometry nodes in the scene graph, as well as the forces applied to all gstPHANToM nodes in the scene graph. gstPHANToM_SCP presents the surface contact point (SCP) in the scene graph. The method getPosition_WC returns a point containing the position of the tip to the PHANTOM in "World Coordinates", which means in a fixed

referential whose origin is the position of the PHANTOM base. getGimbalAngles returns the angles of the pen part of the PhANTOM. These angles are relative to the tip of the pen and are not given in the World Coordinates system. getGimbalAngles returns a vector containing three angles, g1, g2, and g3. These values can be used to transform the world coordinates system into a coordinate system fixed with respect the PHANTOM pen:

Geometry-related classes

The API provides a complete array of geometric structures that can be added to the haptic scene. These structures are inherited from the gstTransform class and can thus be translated or rotated by the calling application via the setRotate, rotate, setTranslate, translate, and setTransformMatrix methods. Transformations can be applied either directly to the objects themselves or to their parent separator. gstTransform also supports Geometric primitive objects as well as objects built using polygonal tri-meshes.

```
cubeSep = new gstSeparator;
cube = new gstCube;
cube->setHeight(size);
cube->setWidth(size);
cube->setLength(size);
cubeSep->addChild(cube);
ff1 = new magneticForce(gstPoint(-size/2,0.0,0.0),amplitude,5,cubeSep);
ff1->boundBySphere(gstPoint(-size/2,0.0,0.0),size/3);
cubeSep->addChild(ff1);
```

Other objects that can be loaded into the haptic scene are shapes described by VRML files (function gstRead VRMLFile) and dynamic objects, whose position and orientation can be changed without any intervention from the calling application (gstButton, gstDial, and gstSlider).

Class gstForceField

Force Fields can also be added to the scene as a child of a gstSeparator. The developer has defined a new class for each kind of force field that must be instantiated. It subclasses gstForceField and overrides the method for calculating ForceFieldforce. This method returns a gstVector, which will be added to the force applied to the PHANTOM if the latter is inside the bounding volume of the force.

```
Class magneticForce : public gstForceField {
Public:
gstVector calculateForcefieldforce(gstPHANToM * phantom, gstVector& torques) {
(.....)
}
}
```

OPENHAPTICS™ TOOLKIT

The OpenHaptics toolkit API has been developed by the manufacturer of the PHANTOM and includes the Haptic Device API (HDAPI), the Haptic Library API (HLAPI), Utilities, PHANTOM Device Drivers (PDD), and source code examples [33].

The HDAPI provides low-level access to the haptic device, enables haptics programmers to render forces directly, offers control over configuring runtime behavior or drivers, and provides convenient utility features and debugging aids. It is best suited to developers who are familiar with haptic paradigms and sending forces directly. This includes those interested in haptics research, telepresence, and remote manipulations. Experts can still use HLAPI and HDAPI in conjunction to take advantage of both SDKs.

The HLAPI is built on top of the HDAPI and provides a higher-level control of haptics than HDAPI, but at the expense of flexibility (in comparison to HDAPI). The HLAPI provides high-level haptic rendering and is designed to be familiar to OpenGL® API programmers. It allows significant reuse of existing OpenGL code and greatly simplifies synchronization of the haptics and graphics threads. The PHANTOM Device Drivers support all shipping of PHANTOM devices.

The HDAPI requires a supported three-dimensional haptic device such as PHANTOM® Omni™ model with installed drivers, and the installed HDAPI. The basic step of HDAPI incorporated haptic rendering is given below in Figure 2.13:

- *Initialize the device*: Both the device and scheduler must be initialized before starting haptic rendering. The first, which calls for initialization in an HDAPI application, is:

HHD hHD = *hdInitDevice* (HD_DEFAULT_DEVICE);

hdEnable (HD_FORCE_OUTPUT) ;

- *Scheduler callbacks*: A scheduler callback that calculates the device position/orientation and commands a force feedback from the virtual object's surface during touch or penetrates the plane is required. The scheduler provides support for high frequency, a high fidelity thread for sending forces at a rate of 1000 Hz, and collecting position/orientation information. It provides consistency between graphic and force rendering. Initialization of the device enables the force by calling the function

  hdStartScheduler ( );

- *Cleanup*: Before exiting the application, all scheduler operations must be ceased. The following functions show a typical cleanup sequence:

  HdStopScheduler ( );

  HdUnschedule (scheduleCallbackHandle);

  hdDisableDevice (hdGetCurrentDevice( ));

On the other hand, the OpenGL API provides graphic rendering, and the HLAPI provides haptic rendering by using a high-level C API. Thus, HLAPI, in accordance with OpenGL, supports geometric primitives such as triangles, lines, and points, along with haptic properties such as stiffness and friction. The architecture of HLAPI incorporated haptic rendering is given below:

In HLAPI, the device should be initialized. This is similar to the HDAPI's use of the function

hHD = *hdInitDevice* (HD_DEFAULT_DEVICE);

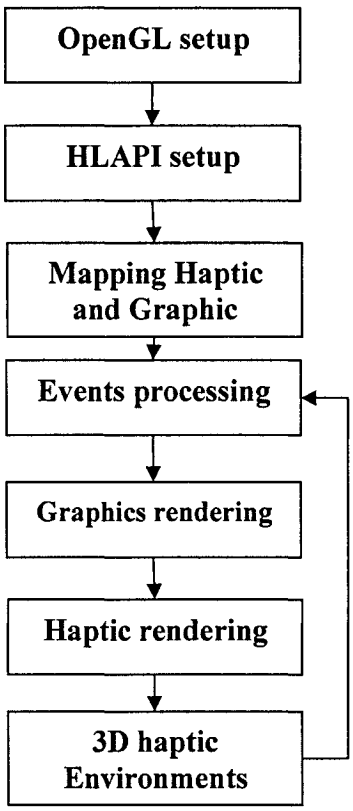


Figure 2.13: The structure for typical HLAPI incorporated Haptic rendering.

After initializing the device, a context that ensures the persistence between frame intervals and haptic rendering, and makes the context current by calling the function, is required. The steps for this process are given below

hHLRC = *hlCreateContext* (hHD);

hlMakeCurrent (hHLRC);

In HLAPI, both haptic and graphic rendering take place alternately or simultaneously in the same thread in terms of accessing the same geometry to render. This avoids the consistency mechanism when haptic and graphic rendering occur in different threads.

Chapter 3 Related Work and Background

3.1 Related Work about Collaborative Virtual Environments

From a high level, we can divide related research efforts into three categories: those that examine the problem from a networking perspective, those that have a human-computer interaction point of view, and those that use predictive techniques. In terms of networking, the Synchronous Collaboration Transport Protocol (SCTP) [36] provides unreliable delivery for normal updates and reliable delivery for key updates. Though it has been shown to improve collaboration under network loss, it is susceptible to jitter and also doesn't perform well for cascading packet loss. Other approaches include a selectively reliable multicast protocol, which is similar to SCTP but uses negative acknowledgements to reduce network traffic [32]. Smoothed SCTP [10] adds strategies for dealing with jitter by creating a small buffer at the receiver side. Incoming updates are hence delayed very shortly and then sent to the application level in fixed rate. This protocol somewhat improves the performance of collaboration due to its jitter mitigation; however, it can't deal with lost update messages or update messages which are delayed longer than the buffer's length of time.

From a human-computer interaction perspective, ***decorators***, which are visual cues typically changing their color with the variation of network delay, are being used to visualize the state of collaboration to the users [16]. For example, a decorator will display the green color when end-to-end delay is less than 50 msec, and the red color when this delay increases to 200 msec. This reveals to the user, visually, the current condition and creates awareness for the user such that s/he can adopt his/her own coping strategy. Both multicolor and 2-color decorators have been examined in Haptic CVEs [37]. Thus, it is a technique based on user's aptitude and their coping strategies in shared object manipulation during closely coupled collaboration; complementary with other network level solution.

In addition to the above approaches, researchers have recently begun to re-examine another method to improve the collaboration quality in shared user spaces: prediction [11, 15] – to mitigate lost updates as well as to predict the future state of the simulation to users. For instance, assume a user moves a virtual object, generating 5 update messages each 10 ms apart. Further assume that update message 3 is lost. In this scenario, message 3 does not need to be retransmitted since updates 2 and 4 are received correctly and one should be able to replace the lost update by predictive techniques. Another example is when two remote users manipulate the same virtual object at the same time. In this scenario, the interaction's direction of movement is crucially important for successful collaboration. Prediction based on history of updates can estimate and visualize the direction of the interaction and aid the collaboration.

The next section describes our work, which combines the predictive techniques with decorators for haptic collaboration. A collaborative haptic application, where two remote users carry a virtual box to a predefined target, is also developed for objective evaluation and as a proof of concept for the proposed technique. The application also uses the existing Smooth SCTP protocol as a network-level solution to carry out various experiments to measure task completion time, and total number of errors in execution time in order to find the efficiency of our approach. Let us begin by examining the proposed system and its prediction method.

Collaborative update messages in an end-to-end communication have strong delay requirements. Like other distributed systems, CVE is susceptible to delays which are the result from the network used to transmit messages, and the processing of those messages at the endpoints. Since the inception of CVE, many studies have been carried out to find out the source [13, 16] and the effect [18, 44] of delays and jitter on synchronized tasks performed by users. It has been suggested that good collaboration environments must have an end-to-end delay of not more than 200 ms for state update messages [31]. In addition, it has been shown that jitter, which can be define as variance in the end-to-end latency (standard deviation of delay) between two users, has a significant adverse effect on the quality of a collaborative session, being even more harmful than delay: a 10 ms jitter can result in a collaboration environment which is almost as bad as one with a 200 ms delay but no jitter [31]. Finally, collaborative state update messages have strict

reliability requirements – that means: update messages are never lost, and always delivered to the end-to-end application in a timely manner in relation to global simulation time.

In this section, we will introduce some similar work in this field done by other teams. Some of them have focus on solutions based on network communications, whether at the transport-layer, the network-layer, or the application-layer in the form of framing of update messages and last one has used decorators to reveal to the users the presence and magnitude of the lag so that the users themselves can better understand the consequences and apply their own copying strategies. We will introduce what these teams have done and the basic concepts and architectures behind their work. The differences between these works and our solutions, as well as the advantages of our solutions, are also presented following the introduction.

3.1.1 Collaborative haptic audio visual environment (C-HAVE)

The collaborative haptic audio visual environment (C-HAVE) [35] is a generic architecture that provides stable, heterogeneous, scalable, and haptic-incorporated collaborative virtual environments. In C-HAVE, users interact with each other as well as other virtual objects, and the user's interactions are transmitted to distributed participants using HLA/RTI, the IEEE standard for distributed simulations, and modeling [17]. C-HAVE also allows users a heterogeneous assortment of haptic devices such as the Sensible PHANToM, the MPB Freedom6S Hand Controller, the Immersion CyberGrasp, and the iFeel mouse, with which users interact with the system. It consists of a network of nodes that usually correspond to computers. Certain nodes have operators who typically interact with the shared virtual environments; however, some such as those who use C-HAVE only for visualization or to supply virtual objects (as in Figure 3.1), are passive participants.

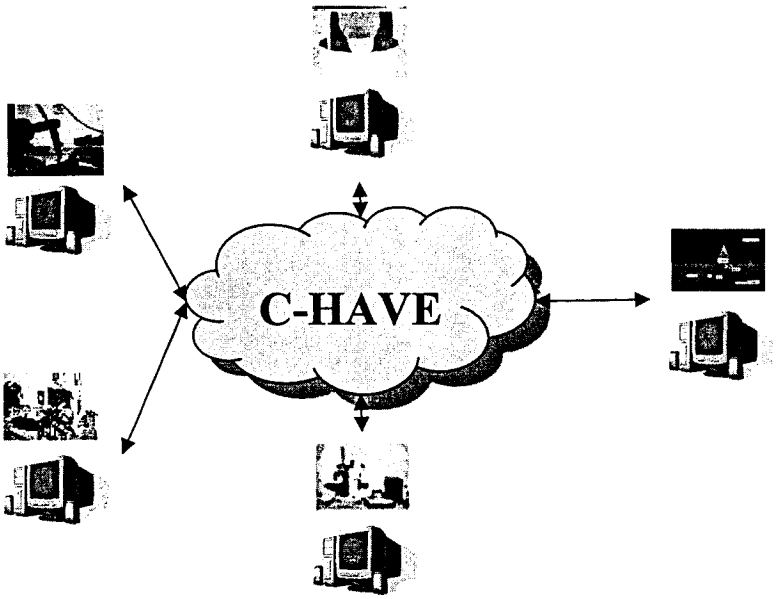


Figure 3.1: Collaborative haptic audio visual environment.

HLA/RTI provides scalability, federation/federates, and object management that deals with services such as Federation Declaration, Object Declaration, Object Ownership Management, Time Management, and Data Distribution Management (DDM). In order to transmit interaction updates to the users, the UDP (User Datagram Protocol) protocol is used for communications protocol.

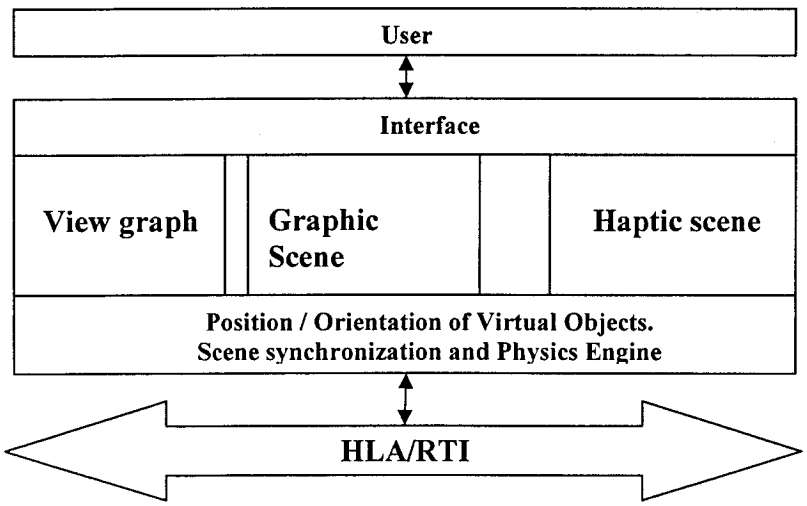


Figure 3.2: Architecture of C-HAVE.

As discussed above, the consequence of network delay in tele-haptic application in C-HAVE, is that the Haptic Real Time Controller (HRTC) compensates for network delay.

HRTC enables real time control and information exchange in tele-haptic by including both hardware and software level solutions. Figure 3.2 shows the architecture of CHAVE.

In order to evaluate the network delay compensation technique in [35], three haptic incorporated prototypes, with or without HLA/RTI, have been evaluated. Although HRTC is used to compensate for network delay, the evaluation results are not satisfactory because of detrimental network delay, jitter, and packet loss. These discrepancies make tele-application instable and impose Parkinson effects on the system [35].

3.1.2 Synchronous Collaboration Transport Protocol (SCTP)

SCTP (Synchronous Collaboration Transport Protocol) is a host-to-host layer protocol and developed at DISCOVER Laboratory, University of Ottawa. It is architecture that supports tightly coupled collaborative tasks to be performed efficiently in virtual environments. The architecture consists of an application-general layer which is mapped into a communication protocol.

To achieve successful collaboration, it proposes to group a series of update messages related to one sequence of interaction of a user with a shared object into one stream. It categorizes the interaction stream into two types: key update messages (critical message) – sent reliably and regular update messages – sent by best effort transport as shown in Figure 3.3.

Key Update	**Update**	**...**	**Key update**	**Update**	...	**Key update**

Figure 3.3: An interaction stream in SCTP

Synchronous Collaboration Transport Protocol (SCTP) assigns specific packet format and reliability issues. It is encapsulated into UDP packets and assumes that the underlying physical network supports IP multicasting. By using IP multicast, it becomes scalable and fast due to the utilization of the low-overhead UDP packets. Figure 3.4 illustrates an SCTP packet and its header fields.

Control bit	Key update	Object ID	Stream numbr	Sequence number	Update message

Figure: 3.4: SCTP packet format

SCTP implements an ACK-based a reliability architecture – key update messages be immediately acknowledged by receiver. This protocol significantly improves the users' ability to collaborate on a network where delay and packet loss is evident. However, it does not address the issue of jitter and cascade packet loss.

3.1.3 Selectively Reliable Transmission Protocol (SRTP)

Selectively Reliable Transmission Protocol (SRTP) is known in networking as a "transport", that is end-to-end connection between distributed hosts. It is successfully used in the RTI/HLA to provide efficient real-time networking for distributed simulation. Although it is not a framework for collaborative virtual environment, SRTP is a communications protocol that can be used to disseminate update information in such environments. It is described as an Internet Draft in 1997; this hybrid protocol has three distinct transmission modes of operation [32] in order to deliver data to hosts who need varying levels of reliability.

- *Mode zero*: for data that does not require reliability. For this mode, SRTP uses a multicast architecture that operates as pure best-effort services.
- *Mode one:* for data that must be received reliably by all members who are in the same multicast group. For this mode SRTP uses a NACK-based reliable multicast protocol, with a "NACK suppression" mechanism as an effort to avoid the NACK implosion problem. In mode one, data send as mode zero, only in mode one, there is a data loss detection technique, which supported by negative acknowledgements for data transmission reliability.
- *Mode two*: uses a positive acknowledgement for data that must be received reliably by a single known member in the group. The supposition is that in this mode, an entity requires reliable communication for each message between itself and only one other entity. For this mode, SRTP uses either TCP or ACK-based

multicast, with non-participating entities ignoring the ACK message. It provides reliability and low latency by avoiding the requirement to open a TCP connection. It operates occasionally among arbitrary members of a large multicast group.

3.1.4 Communication Architecture Based on Updateable Queue Abstraction in Application Layer.

The communication protocol between participants in a CVE system can be explained into three different aspects, such as transport of state update, events, and commands messages [22]. Though exchange of state update massages is the objective of this article, the rest two categories are also important for successful collaboration. In [26], the communication of CVE systems is categorized in three types: state update, command and event. An updateable queue is used to filter state update messages, which are not key state update messages. After filtering, ISTP (Interactive Stream Transport Protocol) packet encapsulate module is used to encapsulate the filtered update messages and will be put to IP layer for transportation, and if the packet is carrying key state update message, command or event, reliable control module will be used to ensure the reliability of transportation. Figure 3.5 illustrates an ISTP packet and it header fields.

Timestamp	Type	Object ID	Stream number	Sequence number	Application pay load

Figure 3.5: ISTP packet format

Thus, updateable queue abstraction to do message obsolescence in application layer and uses interactive stream transport protocol in transport layer to achieve efficient transmission of messages among a large number of participants.

3.1.5 Smoothed -SCTP.

After considering the drawbacks of SCTP [36], Smoothed SCTP [10] was devised to address the jitter problem. In [10], a receiver side buffer was implemented, at the expense of an additional network delay, - jitter can be smoothed. In this protocol updates are sent

the same way as in regular SCTP. However, a timestamp is added in SCTP packet format as shown in Figure 3.6 This timestamp is given by adding to the host current time the time shift then exists between the host clock and a common NTP server.

Control bit	Key update	Object ID	Stream number	Sequence number	Timestamp	update messages

Figure 3.6: Smooth -SCTP packet format.

In Smoothed-SCTP, when an update message is received, it is extracted from the network packet, and acknowledged if needed. On the receiver side, updates are put in a bucket according to their timestamp and are checked the bucket by receiver corresponding to buckets as shown in Figure 3.7. As a result, all update messages, including the update messages that have been generated locally, are processed with a fixed delay, higher than the actual network delay, but constant.

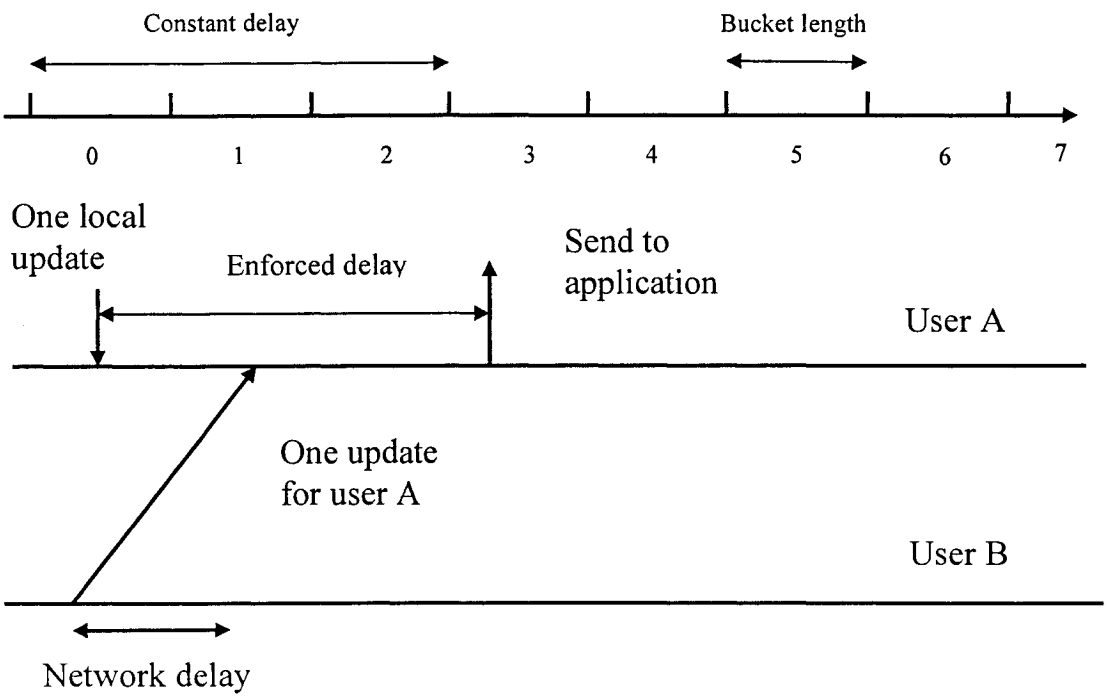


Figure 3.7: Buffering at the receiving end to smooth jitter.

3.1.6 Using Decorators in CVE

Solutions based on network communications, either at the transport-layer, the network-layer, or the application-layer in the form of framing of update messages [10, 22, 26, 36], played significant roll to improve the users' ability to collaborate on a network that presents network delay, jitter or packet loss. However, Solutions based on computer-human interaction point of view which is spawned in [13, 16, 42] is the use of visual ornaments in the representation of an object in order to reveal the network lag to the users, thus user can be aware of the network discrepancies. In [37] Decorators used as visual ornaments, which are the graphical representations to inform the user about the delay or jitter; present visual feedback to the user about the condition of the network. Different mode of color is used to present the network delay increment and decrement. This special technique (decorators) with Smooth SCTP [10] in tele-haptics application improves the users' ability to collaborate on a network that present jitter and delay.

Chapter 4 The proposed Architecture

When users collaborate over the network, certain update messages are created and transmitted between them. These collaborative update messages reflect the actions of the participants as well as the state of the shared objects. Remote instances of the environment can then be synchronized using these messages. Collaborative update messages in a CVE have restricted delay requirements. Like other distributed systems, a CVE is susceptible to delays resulting from the end-to-end propagation delay over the network, and the processing of those messages at the endpoints. Since the inception of CVEs, many studies have been carried out to find out the source [13, 16] and the effect [18, 44] of delays and jitter on synchronized tasks performed by users. As mentioned earlier, good collaboration environments must have an end-to-end delay of not more than 200 ms for state update messages. In addition, jitter, which can have a more adverse effect on collaboration than delay, must also be minimized. Finally, collaborative update messages have special reliability requirements - some update messages can be compensated for if the update message is lost as we shall see, while others must be transmitted reliably.

To model the update messages generated, researchers utilize the concept of "interaction streams" and data transmission protocols [26, 36]. In [36], an interaction stream categorizes updates into two types: key updates (critical message) – sent reliably and regular updates – sent by best effort transport as shown in Figure 4.1. The underlying transport protocol (SCTP in this case) then must ensure guaranteed delivery of key updates for this architecture to work. This protocol significantly improves the users' ability to collaborate over the network; however, it does not address the issue of jitter and cascade packet loss. Figures 4.1 and 4.2 show the header fields of one such protocol.

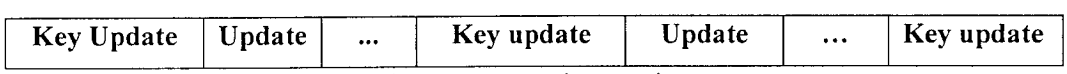


Figure 4.1: An interaction stream

Control bit	Key update	Object ID	Stream number	Sequence number	Application payload (update message)

Figure 4.2: SCTP packet format.

Smoothed SCTP [10] was devised to address the jitter problem. In this approach, a receiver-side buffer is implemented, at the expense of an additional network delay. Jitter can then be smoothed out using this buffer. In this protocol updates are sent the same way as in regular SCTP; however, a timestamp is added to the header to maintain temporal relation between update messages. On the receiver side, updates are put in a bucket according to their timestamp and are checked by the receiver at specific time intervals as shown in Figure 4.3. As a result, all update messages, including the update messages that have been generated locally, are processed with a fixed delay, higher than the actual network delay, but constant.

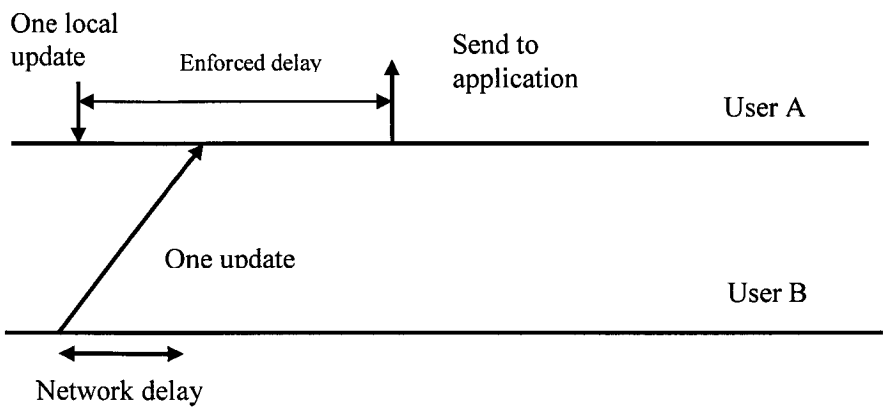


Figure 4.3: Buffering at the receiving end.

In our collaborative Haptic system, all of the above concepts and techniques were implemented. In addition, the proposed prediction method was designed and added to the TOAST [10] framework, described next. Figure 4.4 shows the place of the prediction module with respect to other system components.

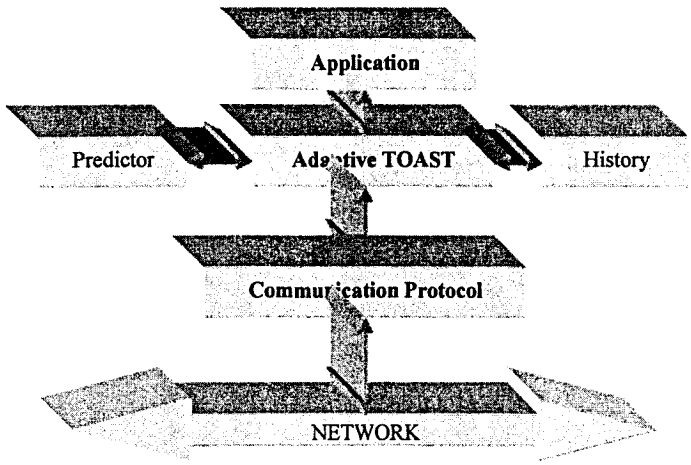
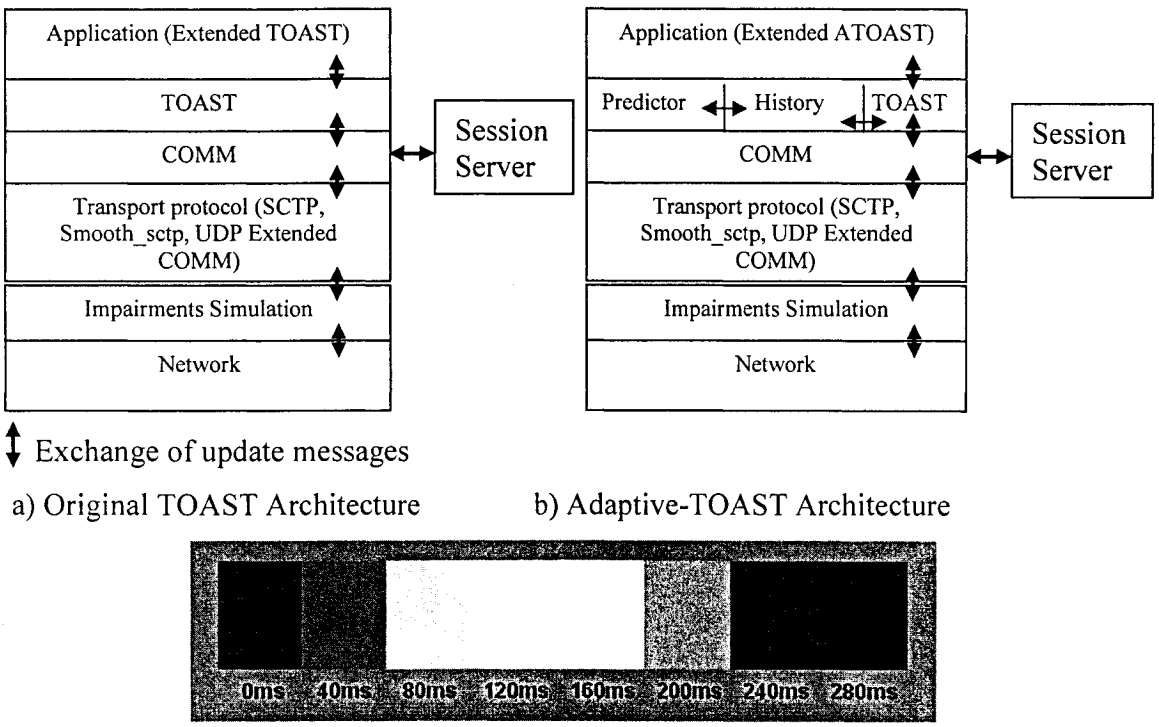


Figure 4.4: Prediction based tele-haptic Architecture

Test and Optimization of Applications in Surgery and Training (TOAST) is developed at the DISCOVER laboratory at the University of Ottawa. It is a unified framework where any transport protocol can be chosen and provides logs of the incoming and outgoing update messages for each host as the amount of traffic generated at the network layer.



c) Color Scheme of the Decorator (0ms left green and 280ms right red)

Figure 4.5: The Architecture of TOAST, Adaptive-TOAST and Color Scheme

Moreover, it also takes care of session issues and is responsible for simulating the network delay, jitter and loss at network layer. The architectures of TOAST and Adaptive TOAST are shown in Figures 4.5.a and 4.5.b.

TOAST can handle interaction streams and load different communication modules. An application needs to extend TOAST to register share objects of the CVE and exchange update messages corresponding to the shared objects. To accommodate our prediction model, we enhanced the TOAST framework with the decorator-based prediction algorithm, which we call the Adaptive TOAST. In Adaptive TOAST, we add two additional elements to TOAST: a history buffer and a predictor that controls a decorator. The predictor is used to improvise for lost update, as well as to predict the delay, jitter, trajectory or state of the object. This predictor also controls the decorator and changes its color according to the currently-experienced network lag, using the values shown in figure 4.5c. The application retrieves updates from the history buffer in fixed rate and updates the screen based on the latest messages.

4.1 The Prediction Model

Each time a new update message is generated by the application or read from the network, a packet is send to the history buffer. As seen in figure 4.2, update messages are composed of an index that identifies which shared object the update message applies to, Stream ID, and Sequence number followed by DOF double-precision floating point numbers - it has been set to six, so that the position of a shared object can be fully described (three coordinates for translation and three coordinates for rotation). Stream ID indicates the ID of the current interaction of the object and sequence number indicates the position of this specific update message in the current stream. From the packet format, we can find out delayed or lost updates for a specific object by checking the stream ID and the sequence number. As with Smooth SCTP, we enforce a small delay in the history buffer for two reasons: to smoothen out jitter and to predict updates. Let us look at an example to better understand the prediction algorithm. Assume an object moves from location (10000, 300, and 1234) to location (9000, 300, and 1234) and creates 4 updates, the third one being a last update and therefore a "key" update. After receiving the first 3

updates, the predictor knows there are more updates coming because the last one received (update 3) is not a key update and hence cannot be the last one. The predictor then guesses the 4th update based on the previous 3rd and puts in the buffer. If update number 4th arrives in time, the predictor simply replaces the predicted update with the actual one. Otherwise, if within the enforced delay the 4th update does not arrive, the predictor increases the enforce delay and sends a message to the decorator instructing it to change color to indicate network lag to the user. In both cases whatever is in the buffer is sent to the application after the enforced delay expires. The same algorithm can apply to an intermediate lost update. Note that the amount of the enforced delay must be small and chosen carefully in order to not cause too much additional delay to the existing network lag. Figure 4.7 depicts the prediction model for the scenario explained above, with white circle indicating normal updates and yellow circle indicating predicated updates. Let us now look at a number of scenarios that can occur for the predictor in the Figure 4.7.

Scenario 1: Application generates 1, 2, and 3 updates and sends them to the receiver. The receiver receives the three updates. The predictor will predict the number four update based on the previous updates and uses the following algorithm (explain in Figure 4.7). Should number four update arrive before the expiry of the enforced delay, it replaces the predicted number four update.

```
If previous update  is key update then
// if update#3 is key update
   future update = previous update
//Update#4 = udate#3
else
predict   future update
//predict update#4
//prediction is calculated from previous update messages
end if
```

Figure 4.6: Future update prediction algorithm

Scenario II: Application generates 1, 2, and 3 updates and sends them to the receiver. The receiver receives the number one and three updates. The predictor will then predict number two and number four updates. Number two is generated by linear predictive method and number four as in *scenario I.* If number two and four come later, they replace the predicted updates.

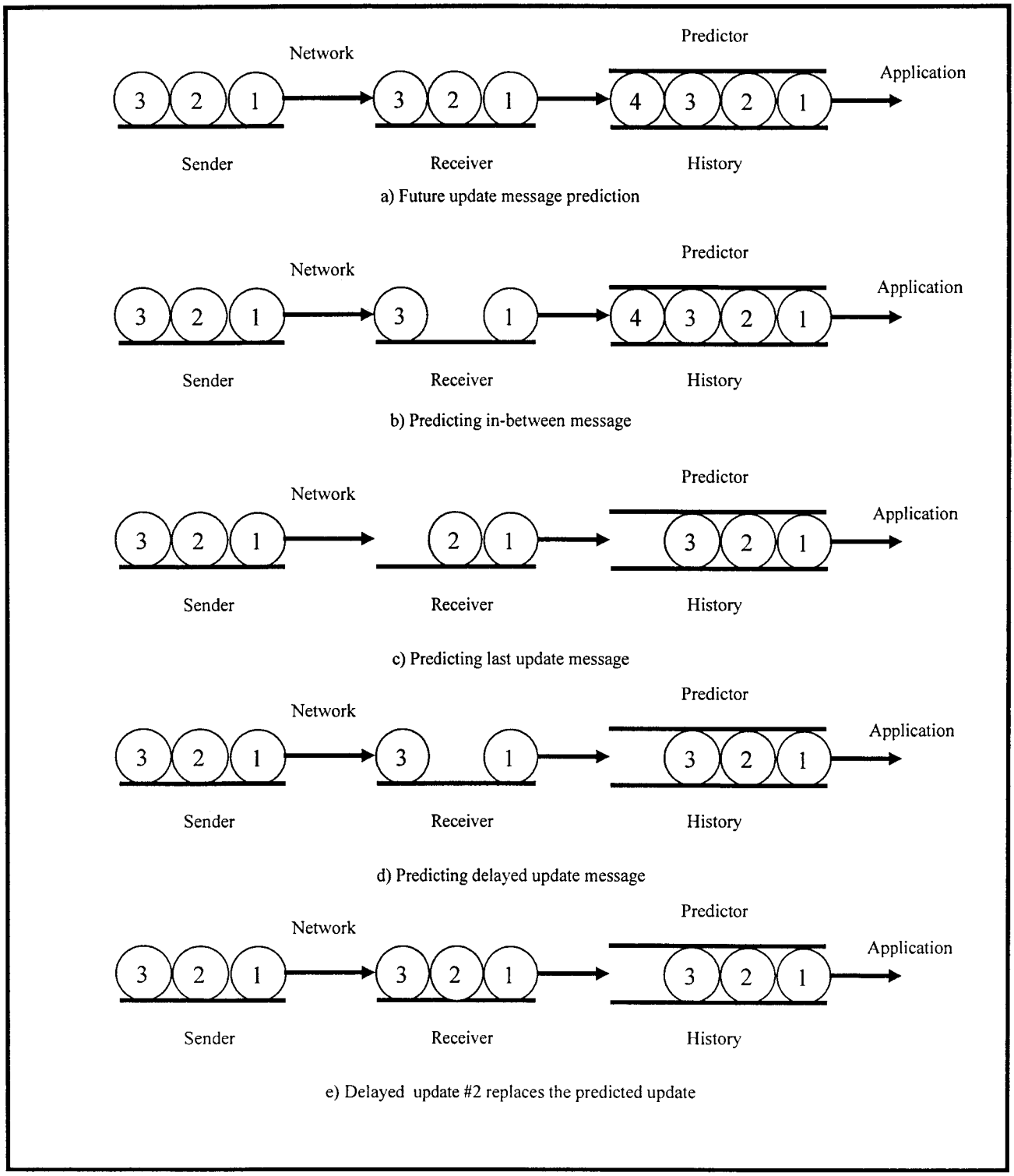


Figure 4.7: Scenarios that can occur for the predictor

Scenario III: Application generates 1, 2, and 3 updates and sends to receiver. The receiver receives the number one and two updates and buffered at the history. The predictor will predict the number three updates. Number three is generated by predictive method as future update prediction algorithm (*scenario I)*. If number three comes later, it replaces the predicted update.

Scenario IV: Application generates 1, 2, and 3 updates and sends to receiver. The receiver receives the number one and three updates and buffered at the history. Update 2 is delayed. The predictor will predict the number two updates. Number two is generated by predictive method as in-between update prediction algorithm (*scenario II)*. If number two comes later, it replaces the predicted update.

While buffering the update messages in the history, the predictor collects receiving time, index, stream ID, sequence number and state of the object of the update message and executes the following tasks.

Jitter decorator: After collecting the consecutive update message's receiving times, the predictor can reveal the delay variation of the transmitting update messages. By changing color of an object according to the amount of time passed between receiving updates, we can implement a jitter decorator.

Direction decorator: The successive states of the object indicate the direction of the movement. This decorator is helpful when an object needs to pass by through either side of an obstacle.

Trajectory decorator: This shows the past and predicted future states of objects, resulting from the history of the past updates. The trajectory helps users predict the potential course of events and hence plan their activities in advance.

Chapter 5 Experimental Environment

The Experimental Environment developed in this thesis is called Adaptive-Transport Layer Optimization for Applications in Surgery and Training (A-TOAST). Its framework is elaborately described in Chapter Four. The main objective of this framework is to provide a complete interface to applications that require haptic collaboration, and support transport layer protocols with or without the prediction module – history and predictor. This Adaptive TOAST is an extension of TOAST [10], which was designed and developed in the DISCOVER Laboratory. The purpose of the experimental environment is to evaluate the effectiveness of the prediction algorithm, which deals with network delay, jitter, and packet loss, and decorator, which non-intrusively creates awareness among the distributed participants. Two applications have been constructed in order to quantitatively demonstrate the effects of the use of haptics, the simulated network, and the proposed algorithm, as well as the detrimental effects of time delay on performance.

5.1 Requirements of Experimental Environment

The Experimental Environment requires providing the following properties:

- In terms of packet format, a precise format for the update messages that can easily be modified if the application's requirements change. Here, update messages are encapsulated into UDP packets that offer scalable and fast IP multicast to the applications.
- Interaction Stream management groups a series of update messages in terms of same sequence of interaction of a user with a virtual object into one stream. Here, the update messages are grouped into two types: key update messages, and regular update messages. The interaction stream management is performed by either application or A-TOAST.
- Prediction module which includes a history buffer and predictor are separate from the TOAST [10] architecture. Experimental environment should support an interface to add prediction module with the transport protocols and the application

remove prediction module from the transport protocols and the application. Those interfaces are used to exchange update messages among the participants. At first, distributed participants able to manipulate a shared virtual object concurrently with no usage of prediction module and decorators. In this case two collaborators share the same VE with suitable network transport protocol. Since, no prediction and decorators to mitigate the network delay, jitter and packet loss, it will be the ideal condition of the evaluation. Finally, distributed participants intend to manipulate a shared virtual object jointly with added prediction module and decorators that help to minimize the network inherent drawbacks. The comparison of both scenarios obviously shows the effectiveness of the prediction algorithm and decorators based solution.

- A network layer impairments simulator is representing the network parameters during the transmission of update messages. The network parameters are delay, jitter, minLossRate and maxLossRate. After receiving an update message at the receiver side, a separate thread handle the packet with simulated network parameters. It provides the required network delay, jitter and packet loss circumstance.
- Logging of exchanged update messages, and application-specific events. For each user, incoming and outgoing update messages have to be logged, along with the time at which the event happened. This way, subsequent analysis of the session is possible.
- Reliable session support, to handle the number of users connected at any time.

5.2 Functions of Experimental Environment

Adaptive-TOAST performs stream handling (if application does not handle), loads the communication module (such as UDP, SCTP, or Smoothed-SCTP) which is chosen from application. Application extends A-TOAST to register share objects of the CVE and exchange update messages with it. Both A-TOAST and application have option to send updates by the methods sendUpdate and sendKeyUpdate. Thus users can select the key update messages and normal update messages or A-TOAST can choose automatically.

Abstract method *receivedUpdate* is callback triggered by the transport protocol for transmitting update messages to application. In A-TOAST Architecture, *receivedUpdate* method is required to implement in application. Receiving and sending updates are logged for exchanging of update messages. The abstract class *Comm* registers the user and gets the participants IP address and port number from session server. Transport protocol is interfacing with A-TOAST using extend Comm. Update message format, Session server and configuration of the network to simulate the impairments are parts of A-TOAST.

In addition to maintain the stream handling, Update message format, Session server, in Adaptive TOAST, we add two additional elements which are history and predictor. The abstract method *receivedUpdate* is implemented in history to create a buffer instead of application. Predictor is using to compensate the lost update as well as to predict the delay, jitter, trajectory or state of the object. Application is retrieving the updates from history in fixed rate and visualizes the network situation, trajectory or orientation of the objects. Local updates are also buffered in the history. Adaptive TOAST loads the communication protocol as well as the predictive algorithm.

5.2.1 Application Interface

Adaptive-TOAST provides the interface between the application which needs haptic collaboration and network protocol. In addition to the above job, Adaptive-TOAST connects and disconnects the prediction module with the application. Any application using Adaptive-TOAST has to extend it to be able to register shared objects of the virtual environment and exchange update messages with it. Doing so, the application inherits the methods *sendUpdate* and *sendKeyUpdate*, and is required to implement the abstract method *receivedUpdate*, which is callback triggered by the transport protocol when passing an update message to the application.

The interaction stream should be managed by both application and Adaptive-TOAST when application is using prediction module with the communication protocol. When the application is interfacing with the communication protocol without prediction module, the interaction stream can be managed either by the application or by Adaptive-

TOAST. The selection of key update is very crucial for successful prediction as well as collaboration. The selection of key update can be achieved by the following methods

- The application always uses *sendUpdate* and Adaptive-TOAST determines which updates among those are to be sent as key updates. This is done by sending every n^{th} packet as key, or sending one key update every t millisecond, n and t being parameters given by the application. This technique is suitable for our prediction algorithm.
- The application determines whether the update message to transmit is a key one or a normal one, and use *sendKeyUpdate* or *sendUpdate* accordingly.

Finally, the class Adaptive-TOAST also logs the update messages when they are sent or received.

5.2.2 The update Messages

UpdateMessage allowed for the design of the format of update messages. A new instance is created every time a new update message is generated by the application or read from a network packet. The composition of Update Messages includes one index, which identifies the shared object to which the update message is applied, followed by DOF double-precision floating point numbers. DOF is a number specified by the framework; this number remains constant throughout a session but can be changed if necessary. In this experiment it has been set to six so that the position of a shared object can be fully described (three coordinates for translation and three coordinates for rotation). An alternative involves exchanging the position of the object in translation and the forces applied to it.

Chapter 6 Tele-Haptic Class of Applications and Experimental Results

6.1 The Collaborative Tele-Haptic Application

Figure 6.1 shows a screenshot of the Haptic application, which consists of a Virtual Environment (VE), Haptic interface, Communication Architecture, history buffer and predictor-controlled decorator. The VE application acts as a visual and tactile interface between the user and the application. A loader reads the descriptions of virtual objects and renders them graphically, while a haptic device acts as the "touch" interface. In the application, a "heavy" virtual box is carried by two remote users who use Phantom [27] haptic devices to interact with the CVE. Specifically, we used Phantom® Desktop^{tm} device which has 6 DOF (degree of freedom) input and 3 DOF output. Phantom gives the feeling of touch and heaviness of the virtual box by force feedback. Force feedback helps us manipulate virtual object with real feelings and make collaboration tasks more realistic. Java 3D was used for rendering the virtual environments and was linked, using JNI (Java Native Interface), with the GHOST (General Haptic Open Software Toolkit) API for haptic rendering. Our objective of this implementation was to carry the box, through obstacles that need to be navigated around, to a predetermined destination by applying both users' interactions simultaneously. Each user has a magnetic hook which should be connected with the knobs on the box. Both participants simultaneously carry the box and put in the destination place. This task requires closely-coupled collaboration. Like real world, if one user pushes too fast and another user pushes too slowly, the box will fall down. If jitter is present and update messages are lost, then it is difficult to complete the task. Thus by this experiment, we can simulate the network lag and measure the performance of our technique. For evaluating our technique, we took the total execution time to complete the task, and the number of errors.

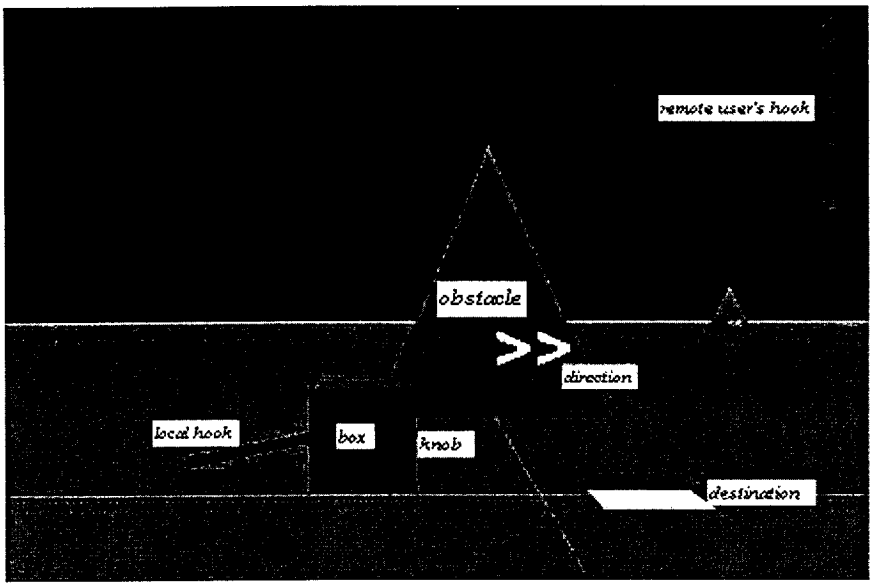


Figure 6.1: Tele-haptic application

A group of five participants were used to conduct this experiment. There were six trials, each involving normal and reverse order experiments to avoid the "training effect". For communications, User Datagram Protocol (UDP), which is an unreliable protocol, Smooth SCTP and SCTP were used. Network delay was simulated by an intermediate node acting as a router that would delay or drop packets. The results are shown in tables 1, 2, and 3, presenting normal order measurements, reverse order measurements, and the average of all measurements, respectively.

Table 1. Results, normal order.

Network conditions	UDP		Prediction based tele-haptics And UDP		Smooth SCTP		Prediction based tele-haptics And S-SCTP		SCTP		Prediction based tele-haptics And SCTP	
	ET	NE	ET	NE	ET	NE	ET	NE	ET	NE	ET	NE
No lag	39.5	52.8	28.0	20.8	24.2	21.9	24.1	29.1	26.9	40.2	19.9	18.5
20% loss	51.2	74.1	34.2	25.1	29.9	27.2	28.9	31.9	37.1	47.0	22.8	22.0
30% loss	70.7	111.8	39.8	37.8	31.7	37.0	34.9	46.1	50.8	93.1	27.5	33.5
10 to 50% loss	64.3	105.2	44.8	47.3	33.0	39.9	33.8	50.4	45.6	90.7	27.3	37.5
50ms delay	34.1	49.9	27.7	24.6	26.2	27.0	29.1	35.7	35.4	55.2	23.0	27.1
50ms delay, 10 ms jitter	49.6	61.8	28.3	32.6	33.2	34.7	36.6	42.3	48.0	77.1	27.9	38.6

Table 2. Results, reverse order.

Network conditions	UDP		Prediction based tele-haptics And UDP		Smooth SCTP		Prediction based tele-haptics And S-SCTP		SCTP		Prediction based tele-haptics And SCTP	
	ET	NE	ET	NE	ET	NE	ET	NE	ET	NE	ET	NE
50ms delay, 10 ms jitter	46.5	67.2	31.2	33.3	33.8	45.6	40.6	65.6	43.3	66.4	31.8	33.2
50ms delay	39.2	54.4	26.0	35.8	31.0	40.4	31.1	41.5	35.1	39.0	22.9	27.5
10 to 50% loss	67.0	110.2	47.2	42.5	31.7	49.3	31.9	60.4	51.9	88.9	29.7	58.1
30% loss	62.2	96.9	42.9	44.6	36.5	46.6	37.9	52.0	59.0	95.8	29.8	44.2
20% loss	45.5	60.3	26.5	32.0	30.1	37.8	31.3	38.8	38.7	60.7	23.5	37.5
No Lag	38.2	53.2	23.7	21.3	31.8	35.0	34.2	44.7	33.8	35.6	19.7	23.4

Table 3. Results, Averaged.

Network conditions	UDP		Prediction based tele-haptics And UDP		Smooth SCTP		Prediction based tele-haptics And S-SCTP		SCTP		Prediction based tele-haptics And SCTP	
	ET	NE	ET	NE	ET	NE	ET	NE	ET	NE	ET	NE
No lag	38.8	53.0	25.4	21.1	28.0	28.5	29.2	36.9	30.4	37.9	19.8	23.0
20% loss	48.4	67.2	30.4	28.6	30.0	32.5	30.1	35.4	37.9	53.9	23.2	29.8
30% loss	66.5	104.4	41.1	41.2	34.1	41.8	36.4	49.1	54.9	94.4	28.7	38.4
10 to 50% loss	65.7	107.7	46.0	44.9	32.3	44.6	32.9	55.4	48.8	89.8	28.5	47.8
50ms delay	36.7	52.2	26.9	30.2	28.6	33.7	30.1	38.6	35.3	47.1	22.7	27.3
50ms delay, 10 ms jitter	48.1	64.5	29.8	33.0	33.5	40.2	38.6	54.0	45.7	71.8	29.9	35.9

Note that ET indicates total Elapsed Time (in seconds, while NE indicates the number of errors. From the results, it is evident that both execution time and number of error is reduced using the prediction-based decorator technique. Using SCTP also improves collaboration, although not as much as the predictor-based technique in this application. An interesting result is that using the predictor together with SCTP seems to yield the best result. This is expected since these methods are complementary. Using Smooth-SCTP with the predictor does not give satisfactory results, due to having two buffers, one for each technique, which increase the delay too much, leading to lower performance.

6.2 Tracheostomy Tele-Surgery

The goal of this procedure is to simulate a surgical act commonly performed in emergency medicine, tracheostomy. This surgery requires very tightly-coupled collaboration between members of the surgery team. In fact, the lack of collaboration can lead to fatal errors. In this scenario, we presented a simple tele-surgery application, defined in figures 6.2 and 6.3, where two surgeons, or trainees, must share tools (a scalpel, two surgical hooks, and a piece of gauze) to cut the skin on a virtual patient's throat, spread it open, and cut inside the underlying muscle layer.

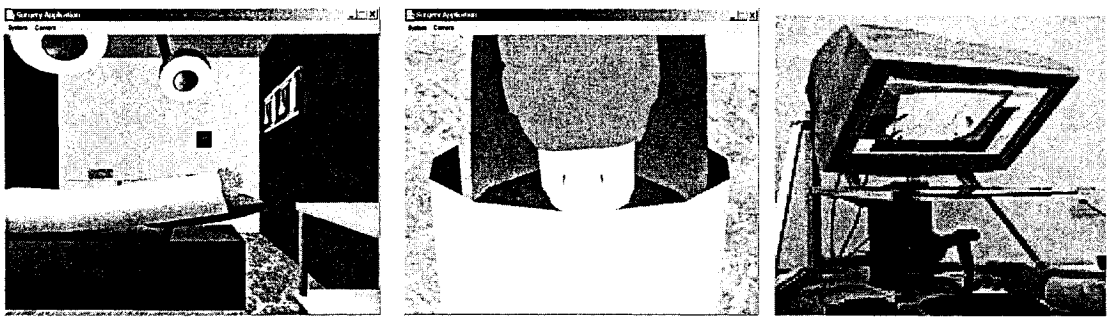
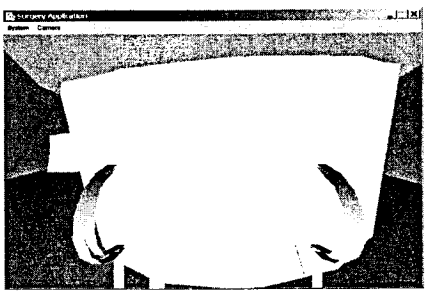


Figure 6.2: Operating room. Side view, Top view and tele-surgery by using haptics

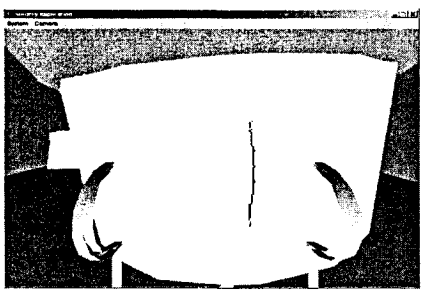
During each simulation session, one surgeon grabs the scalpel and performs a vertical cut while the other surgeon grabs the gauze to remove spilled blood. The first surgeon successively takes both hooks to pull the skin, while the second surgeon performs a horizontal cut on the muscle. Once it has been cut, blood spurts from the muscle. The successive operation is described in figure 6.3.

The above tele-surgery application was put to the test between two users, one is acting as the first surgeon and the other acting as the second surgeon; their respective duties were explained. The success of the surgery depended on close collaboration between the remote surgeons. Collaboration failure happens in two ways:

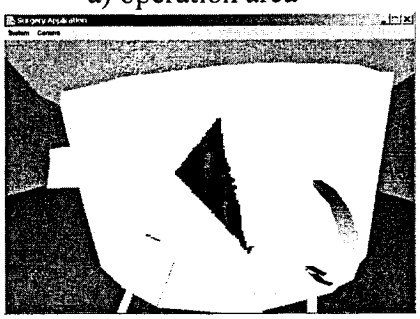
1- The first surgeon cuts the wrong place because he or she does not correctly perceive the position of the surgery, resulting in a great deal of bleeding and the skin being unable to be pulled.
2- The second surgeon cuts the wrong place of the muscle and there is a lot of bleeding.



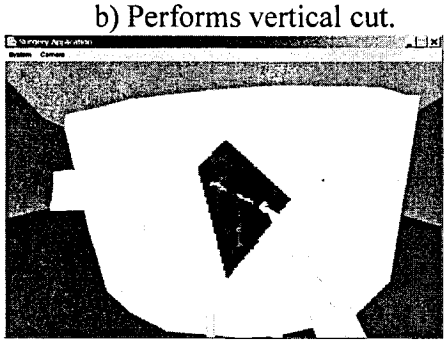
a) operation area



b) Performs vertical cut.



c) Pull the skin.

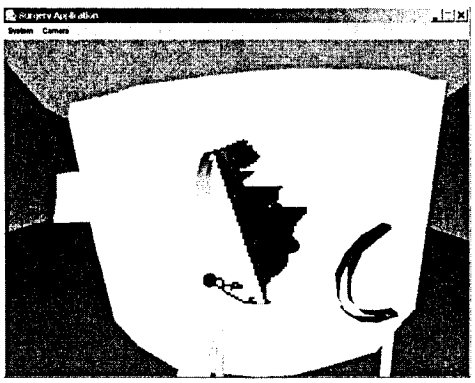


d) Horizontal cut on the muscle

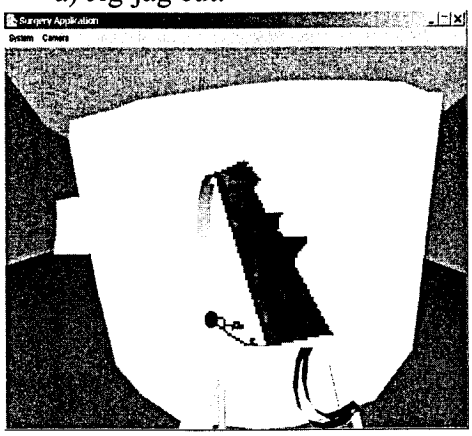
Figure 6.3: Successful collaboration in tele-surgery application.

The unsuccessful surgery is depicted in figure 6.4. Failure of collaboration occurs because of lost updates, jitter, and network delay. Before commencing performance tests in the presence of packet loss, the application was first tested with no packet loss; no failure was observed over sixty-five trials. This is a necessary test to ensure that failures which do occur are not due to the nature of the application but instead to network delay, jitter, and packet loss.

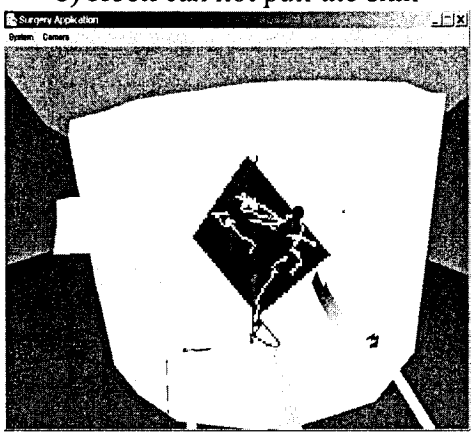
a) Jig-jag cut.



b) Hook can not pull the skin



c) Cut in wrong place



d) Jig-jag cut the muscle.

Figure 6.4: Failure of collaboration in tele-surgery application.

The tests were performed in scenarios in which there was no lag, as well as scenarios in which there were loss rates of 20, 30, 50, 10 to 50, and 50 ms delay, with and without 10 ms jitter. Each test was started with protocol without a prediction model and was later started with a prediction model. We have chosen three protocols: UDP, S-SCTP, and SCTP. The results of the tests are shown in Table 4; it is evident that the occurrence of failure is reduced in prediction-based tele-surgery as opposed to scenarios without a prediction model.

Table 4. Results of the Tele-Surgery application

Network conditions	UDP		Prediction based tele-haptics And UDP		Smooth SCTP		Prediction based tele-haptics And S-SCTP		SCTP		Prediction based tele-haptics And SCTP	
	NOT	NOF	NOT	NOF	NOT	NOF	NOT	NOF	NOT	NOF	NOT	NOF
No lag	65	11	80	2	65	15	80	6	65	4	80	0
20% loss	65	14	80	2	65	15	80	7	65	4	80	0
30% loss	65	14	80	7	65	16	80	7	65	6	80	0
10 to 50% loss	65	12	80	7	65	15	80	4	65	7	80	0
50ms delay	65	17	80	13	65	17	80	7	65	7	80	0
50ms delay, 10 ms jitter	65	21	80	16	65	20	80	7	65	11	80	1

NOT: Number of Trials

NOF: Number of failure

.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

In this thesis, we have introduced a prediction algorithm for tele-haptics that takes into account the key updates in an interaction stream. We enhanced this predictive algorithm with decorators. I have studied inherent network impairments such as delay, jitter, and packet loss in haptic Collaborative Virtual Environments. We took several transport-layer protocols (UDP, SCTP, and S-SCTP) into consideration and used them with the above two applications. We also used these protocols with the prediction algorithm and evaluated their relative efficiency.

Several concepts with respect to haptic collaboration in virtual environments were introduced in this thesis. One was a new perspective that considered prediction for delayed updates in order to compensate for network delay, smooth jitter, and generate the lost updates. Another perspective was to deploy the concept of decorators where the coloring scheme of the magnetic hook presented network delay values; here an arrow indicated the probable direction of the virtual object, and a distinction mark showed the end point of travel. Hence, the solutions can be categorized into three main groups: networking level approaches, the prediction-based algorithm, and user-interaction based approaches. In this thesis, we combined three categories of solutions and applied them to highly synchronous collaborative tele-haptic applications. The results clearly demonstrate that the said hybrid solution, which combines all three methods, gives better results than each method on its own. To the best of my knowledge, no other work has attempted to combine these various types of solutions in the same application and study and report the effect of such hybrid solutions in such a way.

7.2 Future Work

Further study is required to evaluate the effect of prediction with and without decorators. In addition, it might be possible to design non-linear prediction algorithms that work better than the proposed linear approach used here. In the future, we plan to focus on more successful and effective tele-haptic CVE for complex and complicated applications that will provide scalable, heterogeneous, and stable environments. Some important work still remains to be done. I am summarizing some of this work below:

- Verify protocol effectiveness using C/C++ instead of Java.
- The spring-damper models of virtual objects are also the subject matter for network delay and are able to stabilize a virtual environment for greater amounts of network delay. Though the adjustments of spring-damper models are modeled after a constant delay network environment, they are subject to research using the prediction algorithm.
- In this thesis, a linear prediction algorithm is used. Results showed the improvement in collaboration. Further studies may include non-linear prediction algorithms for dealing with unavoidable network delay, jitter, or packet loss include modification of state/orientation estimation techniques. For example, Kalman filters, neural networks, or non-linear regression are my future research topics.
- Interpolation of data can significantly add a delay of several milliseconds. Thus, we believe that it is possible to extrapolate position information instead of interpolating data in order to reduce the delay problem.
- This thesis took a relatively simple haptic device (PHANToM) into account so as to determine how the prediction algorithm is efficiently used with or supported by more complex haptic devices. It is out of the ordinary that Data gloves equipped with force feedback for each finger, or even each joint, have up to 26 DOF. However, there is no theoretical limit to the number of DOF used in the update messages transmitted to users, but packet size is important for transport protocol. Thus, it is a research challenge to transmit in a distributed environment a large

volume of information to users and minimize the inherent network problem. In order to address the data transmission problem, we will investigate the Aggregation/disagregation (A/D) paradigm [4] [5] to ensure consistency in state updates in Tele-haptic applications and the real-time RTI [6] as the middle wire to exchange a huge amount of updates among the participants.

- In this work, it is a very important task to detect collisions between the haptic interface device and the virtual object (rendered surface) in order to reduce computation time and give a realistic response to the user. The calculation of the movement of the virtual box, knife, cotton, or scalpel with which the PHANTOM is in contact requires both a collision detection system to determine the contact surface on the virtual object, and a kinematic model for movement of the virtual objects based on the PHANTOM's motion. To develop an algorithm for collision detection, three recognized approaches, such as point, line segment, or the 3D object shape of the probe, are needed. The prediction-based collision detection algorithm and physics engine are future research topics that could be investigated.

The application of transparent synchronous collaborative environment becomes wider with the development of the Internet and multimedia communication. In order to develop a qualified and practical synchronous collaboration system it is always a challenge to all the researchers working in this field. The work we have done, in this thesis, has contributed to this research, and future proposed work (as outlined above) could reach higher goals to provide powerful, realistic, and generic Tele-haptic Collaborative Virtual Environments for users.

Bibliography

1) F. Arai, M. Tanimoto, T. Fukuda, K. Shimojima, H. Matsuura, and M. Negoro, "Multimedia tele-surgery using high speed optical fiber network and its applications to intravascular neurosurgery-system configuration and computer net-worked implementation," *in Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 878-883, 1996.

2) C. Basdogan, and M. A. Srinivasan, "Haptic rendering in virtual environments", Virtual Environments Handbook, 2001, pp. 117-134.

3) A. Bloomfield, Y. Deng, J. Wampler, P. Rondot, D. Harth, M. McManus, and N. Badler, "A taxonomy and comparison of haptic actions for disassembly tasks," *in Proc. The IEEE Virtual Reality*, 2003. pp 225-231

4) A. Boukerche, and C. Dzermajko, "Scalability and Performance Evaluation of An Aggregation/Disaggregation Scheme for Data Distribution Management in Large-Scale Distributed Interactive Systems," *37th Annual Simulation Symposium (ANSS'04)*, pp. 238-245, 2004.

5) A. Boukerche and A. Roy, "Dynamic Grid-Based Approach to Data Distribution Management", *Journal of Parallel and Distributed Computing*, Volume 62, Issue 3, March 2002, pp. 366-392

6) A. Boukerche, and K. Lu, "A novel approach to real-time RTI based distribution simulation system", *38th Annual Simulation Symposium (ANSS'05)*, pp. 267-274, 2005.

7) W. Broll, "Interacting in Distributed Collaborative Virtual Environments", *Proceedings of the IEEE Virtual Reality annual International Symposium 1995*, pp. 148-155.

8) M. C. Cavusoglu, F. Tendick, M. Cohn, and S. S. Sastry, "A laparoscopic telesurgical workstation," *in IEEE Transactions on Robotics and Automation*, vol.15 pp. 728-739, IEEE, August 1999.

9) D. R. Cheriton, and D. Skeen, "Understanding the limitations of causally and totally ordered communication," *in Proceedings of the fourteenth ACM symposium on Operating systems principles*, vol. 27, December 1993.

10) S. Dodeller and N. D. Georganas, " Transport Layer Protocols for Telehaptics Update Message", *Proc. 22nd Biennial Symposium on Communications*, Queen's University, Canada, May 31-June3,2004.

11) J. Dyck, C. Gutwin, S. Subramanian, and C. Fedak, "High-Performance Telepointers". *Proc. ACM Group 2004*, pp.172-181.

12) N. R. El-Far, S. Nourian, J. Zhou, A. Hamam, X. Shen, and N. D. Georganas, "A Cataract Tele-Surgery Training Application in a Hapto-Visual Collaborative Environment Running over the CANARIE Photonic Network", *Proc. HAVE 2005 –IEEE Intl. Workshop on Haptic, Audio and Visual Environments and their Applications,* Ottawa, ON, Canada, pp. 29-32, October 2005.

13) M. Fraser et al., "Revealing the Reality of Collaborative Virtual Reality", *Proc. ACM Collaborative Virtual Environments*, San Francisco, California, United States, 2000, pp. 29-37.

14) L. Gautier et al., "End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet", *Proc. IEEE Infocom '99*, New York, U.S.A, 1999.

15) C. Gutwin, J. Dyck, and J. Burkitt. "Using Cursor Prediction to Smooth Telepointer Jitter". *Proc. ACM Group 2003*, pp. 294-301.

16) Gutwin et al., "Revealing Delay in Collaborative Environments". *Proceedings of the ACM Special Interest Group on Computer-Human Interaction (SIGCHI) conference on Human factors in computing systems*, Vienna, Austria. 2004, pp. 503-510.

17) High Level Architecture https://www.dmso.mil.

18) K. Hikichi et al., "The Evaluation of Delay and Jitter for Haptics Collaboration over the Internet", *Proc. IEEE Globecom 2002*, pp. 1492-1496.

19) H. G. Hoffman, "Physically Touching Virtual Objects Using Tactile Augmentation Enhances the Realism of Virtual Environments, *in Proc. VRAIS 1998*, Atlanta, GA, USA, pp. 59-63, IEEE Press.

20) IEEE Standard for Distributed Interactive Simulation, Application Protocols, *IEEE 1278-1995*.

21) *IEEE 1278.2-1995*, standard for Distributed Interactive Simulation – Communication Services and Profile.

22) G. D. Kessler, and L.F. Hodges. "A Network Communication Protocol for Distributed Virtual Environment Systems", *Proc. IEEE VRAIS*, pp. 214-221, April 1996.

23) J. Kim et al., "Transatlantic Touch: A Study of Haptic Collaboration over Long Distance", *Presence: Teleoperators & Virtual Environments,* Vol. 13, No. 3, pp. 328-337, June 2004.

24) J. Lanier, "Virtual Reality is the Telephone of the future", in Rucker, R. (ed.) Mondo 2000: A user's guide to the new edge, New York: Harper Collins.

25) J. Leigh, "A review of tele-immersive applications in the CAVE research network", *in Proceedings of the IEEE International conference on Virtual Reality (VR '99)*, Texas, March 1999.

26) C. L. Xu et al., "An Effective Communication Architecture for Collaborative Virtual Environment Systems", *Proceedings of International Conference on Communications Technology (ICCT) 2003*, pp. 1598-1602, 2003.

27) T. Massie and K. Salisbury, "The PHANTOM Haptic Interface: A Device for Probing Virtual Objects", Proc. Symposium on Haptic Interfaces for Virtual Environements and Teleoperator Systems, 1994.

28) M. Meehan, S. Razzaque, M.C. Whitton, and F.P. Brooks. "Effect of Latency on Presence in Stressful Virtual Environments", *Proceedings of the IEEE Virtual Reality 2003*, pp. 141-148, 2003.

29) MPB Communications, Inc., "Live simulated surgery: a "touching" success." http://www.mpbc.ca/mpbc_2004/main_pages/news/2002/6dof.html, December 2002.

30) OpenGL Tutorial http://www.cs.uccs.edu/~semwal/indexGLTutorial.html

31) K. S. Park and R. V. Kenyon "Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment", *Proceedings of the IEEE Virtual Reality 1999*, pp. 104-111, 1999.

32) M Pullen, "Reliable Multicast Network Transport for Distributed Virtual Simulation", *Proc. IEEE Workshop on Distributed Interactive Simulations and Ral-Time Applications (DIS-RT '99)*, Greenbelt, Maryland, pp. 59-66, October 1999.

33) SenSable Technologies, Inc., "Ghost SDK API reference." *http://www.sensable.com/products/phantom_ghost/ghost.asp*

34) SenSable Technologies, Inc., "Open Haptics Toolkit API reference." *http://www.sensable.com/products/phantom_ghost/OpenHapticsToolkit-intro.asp*.

35) X. Shen, J. Zhou, A. El Saddik, and N. D. Georganas, "Architecture and Evaluation of Tele-Haptic Environments", *Proc. The Eighth IEEE International Symposium on Distributed Simulations and Real-Time Applications (DS-RT '04)*, Budapest, Hungary, pp. 53-60, October 2004.

36) S. Shirmohammadi and N. D. Georganas, "An End-to-End Communication Architecture for Collaborative Virtual Environments", *Computer Networks*, Vol. 35, No. 2-3, pp. 351-367, Feb. 2001.

37) S. Shirmohammadi and N. H. Woo "Shared Object Manipulation with Decorators in Virtual Environments" in Prceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04) pp. 230-233, October 2004.

38) S. Singhal, and M. Zyda, Networked Virtual Environments, *ACM Press*, NY, NY, 1999, p. 143.

39) Sun Microsystems Inc., "Java3D API." *http://java.sun.com/products/java-media/3D/.*

40) I. Sutherland, "The ultimate display", *Proc. International Federation for Information Processing (IFIP) Congress*, 2, 1965.

41) United States Department of Defense, Defense Modeling and Simulation Office *https://www.dmso.mil.*

42) I. Vaghi et al., "Coping with Inconsistency due to Network Delays in Collaborative Virtual Environments", *Proc. ACM VRST '99*, pp 42-49.

43) VRML Specification *http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm*

44) D. Wang et al., "The Effect of Time Delays on Tele-Haptics", *Proc. IEEE Workshop on Haptic Audio Virtual Environments and Their Applications*, pp. 7-12, 2003.