National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

Audio/Video Services for a Virtual Laboratory

Yifan Wang

M. Sc., (Chinese Acadamy of Space Technology), 1988

Department of Electrical Engineering

McGill University

Montréal

June, 1994

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

ISBN 0-315-99990-X

Abstract

This thesis presents the design and implementation of a framework for audio/video services for a virtual laboratory. A library of subroutines was developed to support a network video/audio server, capable of performing multimedia interactive presentations in this networked environment. The network video/audio server is an upgraded form of ACME.

In a virtual laboratory, the server's file system can be used to store laboratory lessons, combining circuit simulations such as SPICE or mathematical computations using MATLAB with sounds and video sequences. The clients are able to access the server's file system via TCP/IP connections and to playback a sound or a video sequence, to record a voice message, and to send a voice message or a video sequence to another workstation. The server's workstation and the client's workstations are able to perform real time audio voice communication between each other via the video/audio server. The clients can also send voice commands from the local workstation to the voice recognition toolkit in the server workstation via a TCP/IP connection with the audio/video server for hands-free interactive presentations.

The experimental results indicate that the audio/video performance is suitable for multimedia presentations in a virtual laboratory environment. The suggestions for the future development of the virtual laboratory design are presented.

Acknowledgements

I would like to express my sincere thanks to my supervisor, Prof. Alfred Malowany, for leading me to discover state-of-the-art technologies in the design of a virtual laboratory. I truly appreciate for his guidance and encouragement throughout my study and my thesis work.

I also like to thank the staff of CIM and the Department of Electrical Engineering of McGill for their support.

Finally, I wish to express my appreciation to my fellow students, Gilbert Soucy, A. Hosseinzadeh, Paul MacKenzie, Robert Lucyshyn and many others for their advice and assistance in completing this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Virtual reality (VR) has become a cutting edge technology that allows us to enter a three dimensional artificial world. We can wear the video goggles and data gloves to experience the feeling of flying or swimming, to visit Disney world, to manipulate the objects on the screen with the hands, to pass through a supermarket, or to create 3-D sound. We can also interact with the objects in the 3-D virtual world.

In 1982, William Gilson's science-fiction novel was published. The book described telephone systems that had been replaced by computer networks and people that could fly through data in the computer world into the next century. Now William Gilson's science-fiction can be experienced in virtual reality systems. Virtual reality has evolved as the future of computing.

Virtual reality is that in which people involve themselves to experience some illusionary feelings in a computer world. A virtual world is virtual reality constructed in a natural way for discernment. The virtual environment is the joined elements of a virtual world [Bricken, 1990].

Multimedia combines computing with media, and real-time operation is a major factor. In a multimedia system, we can edit, playback and interact with audio and video on the computer.

The user interface can be constructed as a virtual reality interface, a multimedia interface, or a three dimensional graphics interface.

Both virtual reality and multimedia require system and time integration. Mul-

timedia is distinguished from virtual reality in its interface devices, such as displays, microphones or speakers. Virtual reality uses the interface devices such as the headmounted display which emphasizes the human interface to create illusionary feelings. Multimedia technology and virtual reality technology are coming closer to merging: virtual reality borrows video and image compression technology in its system from the results of multimedia development, while multimedia applies human interface technology to its system from the results of virtual reality development [Latta, 1991].

1.1 Historical Background

Historically, today's Virtual Reality started with the work by National Aeronautic and Space Administration (NASA) to discover a way to train astronauts to remotely operate robotic machinery on the Space Shuttle or on Earth. With the experience from the development of aircraft flight simulators and robotics, NASA implemented a system that allowed the astronaut to interact with the robotic arm using a computer.

Jaron Lanier, founder of VPL Research, invented the term virtual reality to distinguish between the illusionary worlds he wanted to build and the computer simulations. VR is not only a technology that puts the participant inside a three dimensional world, but also an artificial world where its rules can be invented by the designer and the participant feels as if he/she was there [Pimentel and Teixeira, 1993a].

The Helmet-Mounted Display(HMD) and DataGlove are examples of "human interface technology" that enable the interaction between user and computer. The display mounted on the helmet could be a miniature television or flat panel LCD. When we wear the helmet, we can see only the virtual world on the HMD and feel

like we are in a new three-dimensional world. The data glove has fiber optic sensors on it, allowing the images on the HMD to be changed according to the gesture movements of the data glove, and enabling us to feel like we are flying, picking up or displacing objects.

Virtual reality can be divided into two classes. In the first, the users are immensed and get the feeling of participation inside the virtual world. In the second, the users get the feeling of watching the virtual world from outside [Satava, 1993].

The user interface plays a significant role in computing. In the 1980's user interfaces did not have the computational performance of today's interfaces; then only the character interface was used in the computer systems. It was the Apple Macintosh and Microsoft Windows that started to offer users an interface with computational performance. The growing production of graphics accelerator cards has shown how important the interface performance is. For example, engineering workstations are required to have high performance in their graphics and window interfaces.

The future trend in the development of the user interface lies on the improving of the computing performance so that the user interface could be even more user friendly.

1.2 Virtual Reality Systems

The heart of a Virtual Reality System (VRS) is a computer that is connected to the human interface components and that performs the simulation and generates images. The human interface components, such as sensors and effectors, can be utilized as the interface between the participant and the computer [Latta, 1991].

A Virtual Reality Surgical Simulator (VRSS) has been developed for the abdomen

demonstration [Satava, 1993]. The medical student in the VRSS system can experience the feeling of being in the organs, seeing the inside of the esophagus, the stomach, the duodenum, the ampulla of Vater, the bile duct and the gall bladder, and learning their interrelationships. In the VRSS system, the resident doctor may observe a surgical procedure and pick up the virtual tools to carry out a virtual surgery. The simulator of the VRSS can also create various virtual diseases and virtual injuries. This approach to education is much more powerful than the traditional way of education in the sense that the interactive, and visualizing learning processes are related to its instruction elements. Both computer generated graphics and multimedia presentations can be integrated into the learning process [Satava, 1993].

The Defense Advances Research Projects Agency developed network protocols in its Simulator Networking (SIMNET) project. It became one of the first virtual reality products. The network protocols used in the SIMNET project permits the simulated tank and helicopter to interactively operate together in the virtual world in which they participate. This idea was adopted as an IEEE standard for the protocols of Distributed Interactive Simulation (DIS). The Protocol Data Units (PDUs) are sent to the network to implement the interactions among the DIS entities. The PDUs describe the state information of the entities, such as location, velocity, and orientation. The current entity state PDUs will be sent out periodically or whenever the entity state is changed. The dead reckoning algorithms for entity modelling are used to reduce the network bandwidth [Hardy and Healy, 1993].

Voice recognition techniques can be used in the VR system to offer the VR simulation with hands-free operation. The recognition method can be speaker independent with discrete commands, where each action can be commanded by one word. Up to now recognition techniques have not usually been very accurate for the applications of motion guidance. The computations of recognition require a separate system that is parallelled with the VR system to process the voice information and to send out the action command codes [Pimentel and Teixeira, 1993c].

For many years, the development of speech recognition has been aimed at being quick and reliable. Until now, continuous-speech systems, discrete-word recognition systems, speaker dependent systems, and speaker independent systems have been developed. A continuous speech system is normally speaker dependent and each user has to train it for speech recognition. The simpler discrete-word recognition system is usually speaker independent and many people can use the same system with very little training, but the user has to speak word by word with brief stops between them for the system to recognize the speech accurately. The Defense Advanced Research Projects Agency (DARPA-or ARPA) has sponsored BB&N to develop a software package for the recognition of continuous-speech with a speaker-independence feature. The beta-testing of the software on Silicon Graphics INDIGO computers has showed successful results.

A research team in NASA has experimented in using computers with digital signal processors to process sound and to simulate the effect of sound waves on the outer ear [Rousseau, 1993]. This experiment's results indicated that the frequency, amplitude and timing of the digital sound can be modified to reflect the locational variations of the sound source on the actual scene. The modified sound can then be composed into multiple channels and the user can hear the 3-D sound through headphones [Rousseau, 1993].

Combining with the Advanced Training Technologies, a Virtual Evironment (VE) has been studied at NASA's Johnson Space center for the purpose of mission training [Loftin, 1993]. This research focuses on the application of Intelligent Computer-Aided Training (ICAT) to VE, cooperation with other teams in VE, software development for VE maintenance, and a Virtual Physics Laboratory development for learning this NASA training process.

A virtual Hubble Space Telescope (HST) has been developed for training the astronauts in a VE. The virtual HST includes the virtual components that are supposed to be replaced. The VE simulates the supporting device of the Space Shuttle

for sending new equipment into orbit. An ICAT system is used in the VE to provide efficient training for the astronauts who have little knowledge about the HST.

In the VE, a head-mounted display is placed in front of the user's eyes. A DataGlove that is equipped with a location detecting device can simulate the Remote Manipulator System and Man Maneuvering Unit controls. A red "ray" from the finger of the DataGlove can help the user to choose the objects for identification and state changes. All objects related to the training task may be distinguished by selecting them with the "ray" from the DataGlove. The name of the selected object can be given by changing the color of the object and sending an audio message to the user. When an object is selected, the state of the object can be changed by closing the hand. If the user is not sure about the procedure required to complete a task, the ICAT system may issue a help information. If a false action is detected, or if the actions are not sequenced in the required order, an error message is sent and advice on how to correct them is given. The astronauts who are trained with this system will get detailed knowledge about the HST and be able to easily relate the virtual model to the real procedures which will be executed during the mission. This system will enable the controllers at the ground station to interact with the astronauts during the mission, or to replan the mission quickly when there are problems [Loftin, 1993].

The multimedia telepresence technology has been employed in the development of the Mandala Virtual Reality System by the Vivid Group in Toronto. Today's technology has been able to integrate the camera into computer. The camera takes the image of the users and displays it in a virtual world where they can see themselves and interact with images to generate sounds and music and control animation and video, all in real time. Through the camera, the users can use their whole body to interact with the virtual worlds without the need for gloves, trackers or other encumbering devices. Mandala takes the user as the interface, which is different from the conventional interfaces such as keyboards, mice and joysticks.

Mandala VR Systems are currently utilized in many areas, such as teleconferencing, television production, parks, museums, science centers, educations, entertainment and applications for the handicapped. At trade shows, the Mandala VR Systems are used to allow attendees to join the interactive games or to explain their products in an interactive way. The American Museum of History at the Smithsonian Institute uses Mandala VR Systems to enable the interactions between the Information Age Gallery and other locations within the museum. Paramount Pictures have built the simulated movie, "Star Trek: The Next Generation," in which people can actually interact with the scene of the movie. Bell Canada has been also experimenting with the Mandala VR system for teleconferencing in which users in different cities on either end of the connection can interact together [Vivid, 1993].

Today people tend to use Computer Aid Design (CAD) programs, such as AutoCAD, as drafting tools. In fact, we would like to see the simulated reality that we designed. Unfortunately, up to now we were not able to see a very complex simulation because the computer did not have enough computing power and the simulation algorithms were not sophisticated enough to generate realistic simulations. The rendering algorithms of scanline and ray tracing are often used for rendering and animation [Recker, 1993].

The relationship between the image recognition of a human action and the image of the human action in the virtual environment can be combined into a feedback control loop. An image recognition system includes a camera and image processors. Human gestures can be recognized by the gesture recognition system by analyzing the finger position information. The gesture can be translated into application actions. The Visual feedback of the user's actions in the virtual environment is implemented by animating the virtual hand on the computer continuously [Wirtz and Maggioni, 1993].

It has been proposed that the sense of a shared presence can be generated by creating a shared acoustic space with computer audio, using the sounds of the ev-

eryday world. The key clicks and higher level sounds can be included in the shared acoustic VE to indicate the type of work each collaborator is doing, giving the sense that all the collaborators were in the same room. A particular collaborator involved in an activity can be identified from his/her sound location. [Digiano and Buxton, 1993]

It was mentioned in [Thomas and Stuart, 1993] that the virtual auditory display for air traffic controller has become a very interesting field recently. In order to improve the situational awareness of the air traffic controller, audio icons for the incoming flights are placed in space on the virtual auditory display. The speech from the pilot of an incoming flight can be heared from its audio icon position.

Speech recognition technology can be used in a VR system that is shared by several known users, since the technology can allow the system to identify the known voices differences among the users in this shared VR system. Initially the system-independent recognition carries out the user's voice interactions and subsequently uses these initial words to identify which user is currently on the system. Later speaker-trained data could be accessed for more accurate recognition. Voice recognition for speaker verification could replace the login and password schemes and could identify which user is operating the system, it could give users their own virtual environments that is customized with gestural language, glove calibrations, choices of presentations and non-speech audio cues. In the VR system, the user may grab an object while simultaneously saying "enlarge" or "rotate" or "delete." With this scheme, pauses in the speech are natural and discrete. It would allow the recognizable vocabulary to be much larger and the recognition more accurate [Thomas and Stuart, 1993].

VR can be a collaborative space where people could design with gesture and vision, while simultaneously discussing things with each other using speech. One of the important issues in multi-user collaborative VR systems is to distinduish between human-human communication and the command language. In non-VR sys-

tems, the direction the user is facing is used to distinguish system commands from other speech. In this system, the speech signal is received by multiple microphones and the amplitude of the speech signal is used to determine the direction in which the speaker is facing. In VR systems, the motion detector is used to determine the user's position. A graphic representation can be used in the virtual world, and user speech in a specified direction is considered as system command. If speech is used both as command language and for human-human communication in a multi-user system, it should send only human-human speech to other users, so that the system commands spoken by the users are removed. The user's position can be used to identify if a system command is being spoken. The system would not transfer the speech based on a specified location to other users [Thomas and Stuart, 1993].

Psycholinguistics have long been interested in the interpretation of human gesture (movements of the arms, hands or head [Neapoulous et al., 1986]. Now the gesture has also found its place in VR systems for humans to interact with the VR system. McAvinney suggested that 3-D applications, such as 3-D CAD, need the 3-D inputs such as gesture [McAvinney, 1990]. Sachs introduced the 3D tracking of a stylus to a referrence frame in the gesture input system [Sachs, 1990]. Bolt presented an application that used pointing and speaking to manipulate figures of ships on the screen [Bolt, 1985] [Bryson and Gerald-Yamanski, 1992].

In virtual reality, interactions can be achieved by visual feedback [Bohm et al., 1992] , 3D sound [Chapin and Foster, 1992], force and touch feedback [Shimoga, 1992] [Burdea and Lanangrana, 1992]. Virtual reality will become part of the high end workstation [Mogal, 1992]. The tools for transmiting feedback signal to the user hand are under investigation [Iwata, 1990]. Serveral prototypes have been introduced consisting of gloves with touch/force feedback [Cutt, 1992], [Stone, 1991]. A patented force feedback structure provides user with feeling of the virtual objects [Burdea and Zhuang, 1992]].

1.3 Multimedia and Networking Technology

Multimedia is popularly identified with combined presentations, including video, images, motion or animations with sound. It involves computer applications to both spatial media, such as printed materials, graphics and fax; and temporal media, such as video, sound and telephone. Television station operations are similar to computer network operations. Based on the user actions, multimedia presentations can run continuously and change dynamically. The network and its connectivity are involved in the environment design, if multimedia technology is used on a LAN/WAN. The protocols for performimg multimedia communication on the network are included in the operation models. The media peripherials and media inputs at the workstation are controlled by the operating system.

[Szuprowicz, 1992] presents various multimedia applications, including:

1. Home multimedia products involving the presentations of movies, cable TV broadcasts, music, books and magazines.
2. Desktop videoconferencing consisting of an attached video camera and desktop computer with an interactive multimedia facility that is part of a network. It combines teleconferencing and networking with multimedia computing.
3. Sales and marketing applications to replace many traditional paper documents such as travel tickets and finacial statements.
4. An electronic brochure that can be delivered in a single diskette with interactive multimedia programs, including photos, graphics, animation, digitized sound, text and data.
5. An electronic catalog ordering system that enables the customer to choose and compare the products on the screen and immediately send an order to the distributor via modem.

6. Multimedia merchandising systems that can show the flooring samples to the home decorators.

7. Travel-related applications that have the visual presentations of locations and attractions for reservations of flights and hotels.

8. Real estate multimedia applications that allow home buyers to browse through sample houses. The HomeVision interactive multimedia system developed by Denmark's largest real estate chain is an example.

9. Pharmaceutical promotions that can challenge physicians at medical meetings in an interactive game. The example is the interactive multimedia system developed by SmithKline Beecham Pharmaceutical.

10. Financial service applications that can offer interactive multimedia service in banking, insurance and travel without bank employee assistance. For example, IBM is collaborating with a major bank and multimedia creators to develop the "Bank Branch of the Future".

11. An insurance claims liability assessment that deals with Claim Handling or Homeowner Loss Handling. It is also used for screening of job applicants. The Claim Performance System developed by Allstate Insurance has been installed in 230 locations.

12. Industrial multimedia applications to provide service personnel with maintenance manuals or expert diagnostic systems with visual demonstrations and voice explanations of the correct procedures.

13. A "just-in-time" interactive learning multimedia system that can access its knowledge base to demonstrate how to disassemble a machine, repair, and make it work again. The system can formulate logical questions and gather the responses for automatic troubleshooting.

14. Unattended interactive merchandising or electronic retailing, that can display products, play product-related games, interactively design customized prod-

ucts. It consists of a computer with touch-screen display, a videodisk player and a printer.

15. Multimedia applications in goverment services, such as an interactive kiosk developed by the California Department of Motor Vehicles for renewing drivers' licenses.

16. A corporate training interactive multimedia system that allows users to progress at their own pace. It can be used in flight crew training, security and safety training and medical training.

It was predicted that multimedia will be realized offering 30 frames/second full-motion VCR quality video in 1994, NTSC/PAL quality in 1995, and HDTV quality in 1996. The operating platform will be realized with a common platform-independent operating system in 1996. Video compression will be realized with MPEG-2,3,4 in 1995, and Wavelets(research stage) in 1996. The storage will be realized with 500 Mbyte Worm in 1994, 128-Mbyte magneto-optic (R/W) in 1995, 500-Mbyte magneto-optic (R/W) in 1996. The LAN will be realized with Ethernet, token ring (100 Mb/s) and FDDI (500 Mb/s) networks in 1994, Isochronous networks in 1995, and Asynchronous Transfer Mode (ATM) switching networks in 1996 [Cole, 1993].

The application of multimedia technology in UNIX with Ethernet is very difficult with the currently available systems [Coulson, 92a]. Continuous media (video and audio) require real-time performance. The synchronization problem for continuous media has been discussed in [Little and Ghafoor, 1990] [Steinmetz, 1990]. It has been proposed to have two separate channels (one for data and another for control) for the simultaneous real time data delivery of each information type [Mitsuhashi et al., 1987]. In order to obtain a synchronization control, the multimedia data segmentation method has been disscused but is not yet feasible. Currently several sites on the Terrestrial Wideband Network have been operating the experimental packet teleconferencing system [Casner et al., 1990]. The demand for connection management abstractions in the multimedia teleconferencing systems are

emphasized [Leung et al., 1990] [Bates et al., 1990] [Vin et al., 1991] [Lang et al., 1988] [Nicolaou, 1990]. [Little and Ghafoor, 1992] warn that the scheduling approach to delivering multimedia service, such as compressed video, to the home via the 1.5 Mbits/sec link could be a problem if there is other traffic on the link [Rosenberg, et al. 1991]. The connection request for a multimedia service through accessing a shared network relies on the availability of resources [Lazar, et al. 1990].

An experimental VideoWindow system has been developed that realizes the video and audio communication with visual help and shared computer applications, between areas where people can congregate and interact [Addeo and Weinstein, 1992]. It illustrated that in the future people will conduct interactive teleconferencing in a mutimedia environment on a large screen. Composing the videosources produced at different locations or at different times into a single scene has been studied [Little and Ghafoor, 1991]. Multimedia technology has been applied to the research and development of multi-point video conferencing systems, video editing/publishing systems and advanced workstation displays. It was proposed that video can be adjusted imperceptibly by dropping or replaying a frame as necessary, and that voice can also be adjusted imperceptibly by adjusting silent periods. These approaches have been employed in packetized voice applications as early as 70's [Weinstein and Forgie, 1983].

[Demers et al., 1989] introduced a time-stamp algorithm called the "fair queueing" algorithm for network scheduling and guaranteed traffic that multimedia applications require [Hyman et al., 1991] [Clark et al., 1992]. With circuit switched or packet switched techniques, a Dynamically Adaptable Multiservice System is being developed to offer both delay sensitive telephony services and time-independent data services [Cordes et al., 1992]. A multimedia object that is stored on a network file server may consist of media components that may be created by various sources at different times. In order to display this multimedia object, it requires simultaneously placing all the media data (a video frame or an audio sample) that constitutes a multimedia object on a relative time scale. With this technique, the UCSD Mul-

timedia Laboratory is developing a digital multimedia-on-demand storage server [Rangan et al., 1992].

[Lee and Kung, 1992] proposed a high speed priority buffer-insertion ring network to decrease the transmission delay for real-time service, where every packet generated by each workstation has been assigned a pre-defined priority byte and an additional link is added [Chao et al., 1989]. A multimedia application has been developed using a hybrid multimedia testbed on the ATM network demonstrating an employee relocation service in which the employee can setup a collaborative video conference call to select a new home [Gutfreund et al., 1991]. Distributed multimedia computing platforms with high speed switching connection are used for active multimedia compound documents, distributed scientific visualization and HDTV signal distribution[Gutfreund, 1991]. The LyricTime prototype multimedia system was developed to investigate adaptive information filtering at Bellcore.[Loeb et al., 1992]. [Heinrichs, 1992] proposed offering enhanced functionality to the existing the TCP/IP network for multimedia applications.

1.4 Thesis Overview

This thesis presents the design and implementation of a framework for audio/video services for a virtual laboratory. In the virtual laboratory, the server's file system is used to store shared training or teaching material such as a circuit fault diagnosis program, laboratory demonstrations based on simulations or computations using software SPICE [1] or MATLAB, and including sounds, graphic animations, and video sequences. Via a TCP/IP connection, the client's workstations are able to access the server's file system and perform teaching sessions.

[1] SPICE is a popular simulation tool for electronic circuit analysis.

This thesis began with an overview of state-of-the-art technologies in the areas of virtual reality systems, multimedia applications, and their historical backgrounds. A number of virtual reality systems that have been developed, or are currently undergoing development, are surveyed.

Chapter 2 introduces the audio/video server concept and Abstractions for Continuous Media (ACME), which is used in the implementation of the virtual laboratory framework by offering a network server for the clients to access the server's file system via the TCP/IP connection and for performing the video/audio presentations.

Chapter 3 presents the functional description, the harware and software architecture of the virtual laboratory design. It describes the design and implementation of the virtual laboratory sub-routines that can be called from the virtual laboratory main routine. A test demonstration is presented which provides a user interface window on the screen. With the user interface window, users can press pushbuttons to playback a sound or a video sequence, to record a voice message, and to send a voice message or a video sequence to the server's workstation for interactive demonstration. The server's workstation and client's workstations are able to perform real time audio voice communications between each other via the continuous media (Video/Audio) server on TCP/IP connections. Via a TCP/IP connection, clients can also send voice commands from the local workstation to a voice recognition system connected to the server workstation for hands-free interactive presentation capability.

Chapter 4 presents the implementation of the test user interface window with the InterViews interface builder. It also presents the network video/audio server upgrading. The experimental results show the performance and the capacity of the video/audio network server. Future developments of the virtual laboratory with compression/decompression and high speed networking technologies are suggested for improving the video/audio performance. Other recommendations for future

work are also discussed.

Finally chapter 5 presents the conclusions of this virtual laboratory study which explored the capabilities of using the TCP/IP network for the video/audio presentation, voice communication and hand-free interactive demonstration in a cooperative virtual environment.

Chapter 2 TCP/IP Network and Video/Audio Server

In the proposed virtual laboratory, a server workstation is connected to several client workstations via the TCP/IP connection. We need to explore the existing TCP/IP network for operating the virtual laboratory and performing continuous media applications.

2.1 TCP/IP Network

The TCP/IP protocol deals with the computer network communications. It is necessary to break the communication task up into modules or entities because of the complexity of the communications. Other programs or processes may also communicate with these entities. In turn these TCP/IP entities use other entities' services. The TCP/IP protocol involves three communication agents: processes, host, and networks. Processes on hosts are the fundamental entities for communication. The multiple simultaneous processes are supported by the hosts. The hosts are connected to the networks across which the processes communicate among each other. In the protocol, there are multiple layers namely network access layer, internet layer, host-host layer and process/application layer.

The network access layer constitutes the link between a communication node and an attached host. A communication node is a network boundary sending or receiving data between the two hosts connected to two nodes. The network access layer routes data between hosts attached to the same network. The flow control,

error control, and quality of service between hosts are regulated in this layer.

The internet layer contains the procedures for data to traverse multiple networks between hosts. It offers a routing function and is usually implemented within hosts and gateways that connect two networks and whose primary function is to relay data between networks using an internetwork protocol.

The host-host layer contains protocol entities for delivering data between two processes on different host computers. A protocol entity at this level provides a logical connection between higher-level entities to exchange data between the processes. It usually contains the services of error and flow control and handles the control signals not related to a logical data connection. This level requires a reliable connection-oriented data protocol, a datagram protocol, a speech protocol, and a real-time data protocol.

The process/application layer contains protocols for computer-to-computer resource sharing and terminal-to-computer remote access.

The last three upper layers of the architecture contain the following protocols:

- Internet Protocol (IP) in the internet layer: Provides the end systems with connectionless communications across one or more networks. It assumes that the networks are not reliable.

- Transmission Control Protocol (TCP) in the host-host layer: A reliable service for end-to-end data transfer.

The IP offers a datagram service. In datagram service, each packet is processed independently between stations on a connectionless logic link. In a connectionless logic link, data are sent to another node without prior planning of setting up a connection and each sending node determines the next receiving node. Beginning at the sending station, the process sends a datagram between two stations on different networks. The IP module in the station creates the datagram with the address of a global network and sets the destination on another network. First it adds to

the datagram a header designating the network that contains the router's address. Then the packet passes through the network to the router. The router recovers the original datagram by processing the packet and analyzing the IP header to determine if this datagram contains control information for the router, or data for the next station. The header contains the protocol identifier, length indicator, version, Process Data Unit (PDU) life time, flags, type, PDU segment length, PDU checksum.

In TCP protocol, the data that will be transmitted by TCP is stored in specific buffers. By signaling urgent data, the destination user is informed of the significant data that is in the upcoming data stream. The destination user can then decide on an appropriate action.

Following this brief introduction of TCP/IP networks, we will now present the network video/audio server on the TCP/IP network. [Stallings, 1991]

2.2 Network Video/Audio Server Overview

In order to implement the Virtual Laboratory using a client/server model, a network server is needed to perform audio/video communication between file server system and client workstation. The client/server model combines the use of centralized facilities such as a file server with a distributed set of applications running on a separate client workstations. The network server called Abstractions for Continuous Media (ACME) [1] was adopted for our implementation of the virtual laboratory. The ACME abstractions include physical devices (PDEV), logical devices (LDEV), compound logical devices (CLDEV), and logical time systems (LTS). The physical devices correspond to the audio/video I/O devices, such as the micro-

[1] The ACME material presented in the following sections of this chapter includes selected extracts from [Anderson et al., 90a], [Anderson et al., 90b] and the original ACME source code.

phone, speaker and video display of the workstation. The logical device represents virtual I/O devices that read or write data to the buffers related to the physical device via network connections. The compound logical device represents logical device sets. The logical time system represents the timing setup for the compound logical device. These ACME abstractions will be elaborated in section 2.3.

Figure 2.1 illustrates an example of the ACME operation environment. Here the application running on the workstation is shown using two ACME servers (one local and one networked) and one X-window server. The window server deals with the discrete input or output devices, such as a mouse or a keyboard. ACME server deals with the continuous input or output devices, such as audio or video. With the ACME server, a "strand" or a "rope" of continuous media (CM) data can be transfered to a client workstation or to another ACME server, via TCP connections called CM connections. The "strand" is a stream of continuous audio or video data encoded in a specified encoding algorithm. The "rope" is an interleaving of multiple strands of data such as audio and video. Remote procedure calls (RPCs) that are sent on a TCP network link are used by the client to interact with the ACME servers.

Another example in Figure 2.2 shows the relationship between the ACME SERVER and the File Server system. The ACME server builds a network connection for each workstation to remotely access the file server system for continuous media (audio/video) services. A client application may then send requests to the ACME server at a workstation for a playback of an audio/video file that is stored in the remote file server system, or to record an audio file to the file server system. To respond to the request, the ACME server will send a command to the ACME library in the file system via TCP connection. The ACME library functions will process the ACME server's commands, send a reply command to the ACME server and transfer the data file to the buffers of the local workstation via the TCP connection. Then the ACME server will manage the audio/video input or output processes. In doing these, ACME calls InterViews library functions to provide I/O multiplexing features for ACME clients.

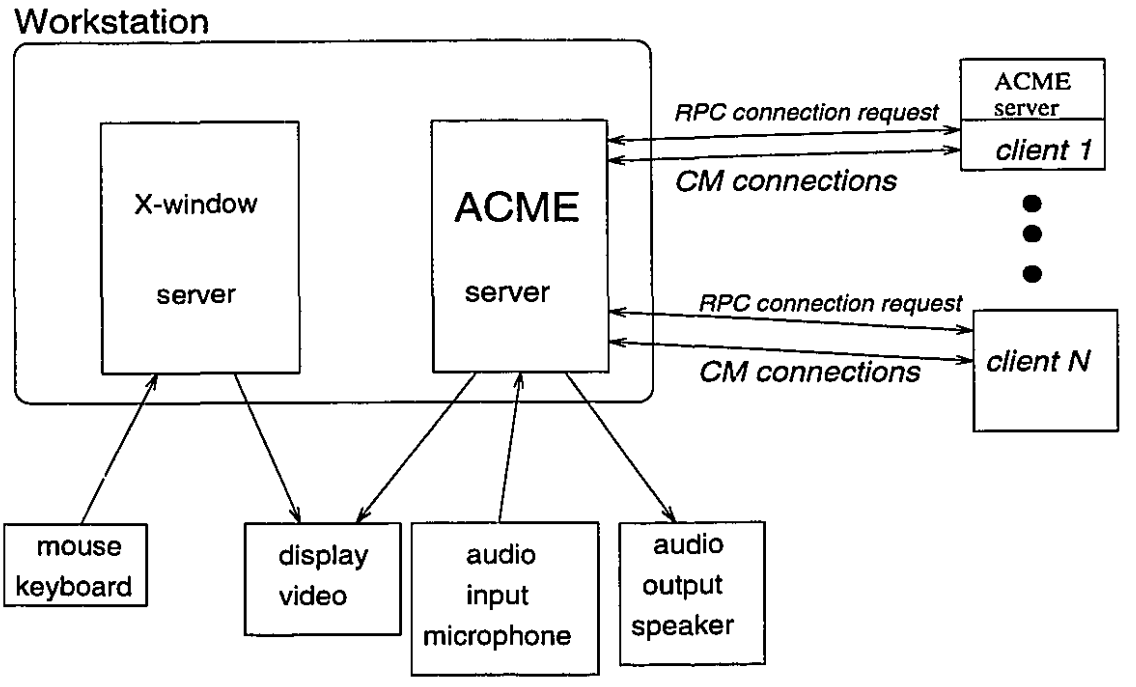

Figure 2.1: ACME Operation Environment

ACME can also communicate with other ACME servers in processing audio/video connections to peer workstations.

ACME is UNIX dependent because of the system call is implemented using UNIX services such as sockets and remote procedure calls in ACME. It is SPARC dependent because of the context switching and stack initializations. It is also SPARCstation dependent since it needs to handle SPARCstation I/O devices. ACME could ported to any other type of platform (which supports InterViews) by modifying the operating system calls, context switching, stack initializations and handling the given machine's I/O devices.

With ACME, the audio sound can be recorded from a microphone that is connected to the SPARCstation's 8 KHz 8-bit μ-law audio input with 1024 samples per block. The audio can be played back to the SPARCstation's built-in speaker at a sampling rate of 8K or 4K samples/second and 256 samples per block. Currently ACME only supports uncompressed monochrome video data played back on the monochrome monitor of a SUN SPARCstation at 15 frames per second providing

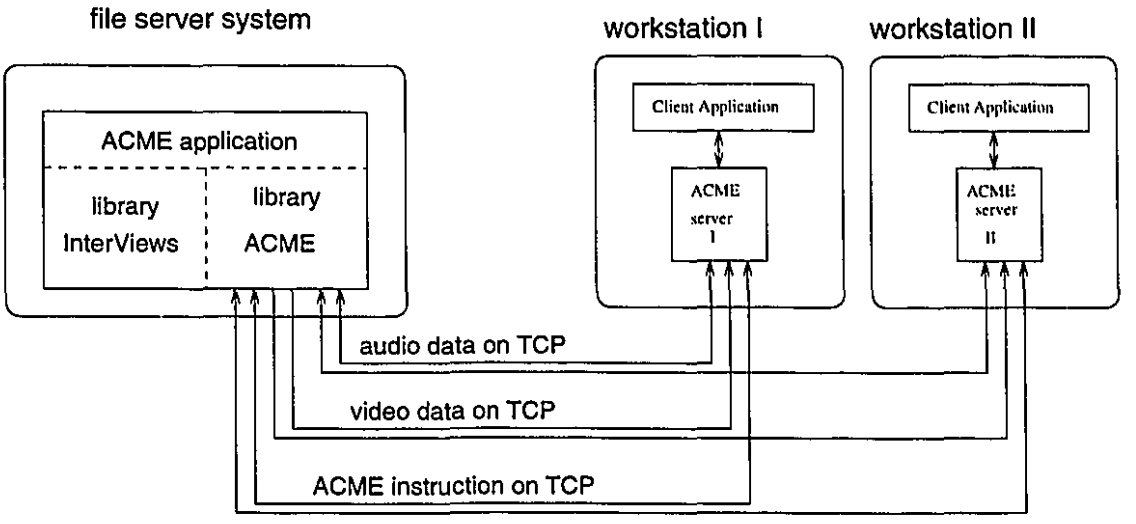

Figure 2.2: ACME SERVER and File Server

the network is not overloaded by other traffic.

2.3 Software Architecture of the Network Video/Audio Server

The ACME server program was written in C++ using the GNU g++ compiler. Before describing its architecture, some features of the object oriented programming will be presented. In order to take full advantage of the power of C++ in a complex design, an object oriented programming methodology is recommended which exploits abstraction, encapsulation, and class hierarchies. The traditional procedure-oriented solution carries out a series of steps, i.e. an algorithm. In the object oriented approach, the program describes a system of objects interacting. Abstraction is the process of ignoring details and concentrating on the important characteristics. For example, a C program is more abstract than an assembler version, making it clearer and easier to understand. Procedural abstraction permits ignoring details about processes while data abstraction ignores details of how a data type is represented. Object oriented languages combine procedural and data abstraction

in the form of classes. Although a class describes everything about a high level entity at once, using an object of a class can ignore the built-in types contained in the class and the procedures used to manipulate them. It is still possible to simply write procedural code using C++.

Encapsulation is the process of hiding the internal workings of a class to support or enforce abstraction. Here we distinguish between a class' interface which has public visibility and its implementation which has private visibility. The class' interface describes what a class can do while its implementation describes how it does it. The user (programmer) views an object in terms of the operations it can perform, not in terms of its data structure. An important feature of object oriented programming is the ability to define a hierarchy of types. We can define one class as a subtype or special category of another class by *deriving it* from that class. If a class is derived from an existing one, inheritance will allow code reuse. A class hierarchy designed for code sharing has most of its code at the base classes, near the top of the hierarchy, so that the derived classes representing specialized or extended versions of the base class can all re-use it.

Another inheritance strategy is for the derived class to keep the same interface as the base class but to perform different things with the same functions, by providing its own code for those functions. Here the derived classes share the same interface making it possible to manipulate them as generic objects. The object oriented design differs from the traditional structured top-down programming, procedural decomposition approach. Instead, the problem is analyzed as a system of objects which interact. The basic questions become "what are the objects or active entities" for this program. The recommended steps of an object oriented design become

1. identify the classes
2. assign attributes and behavior
3. find relationship between classes

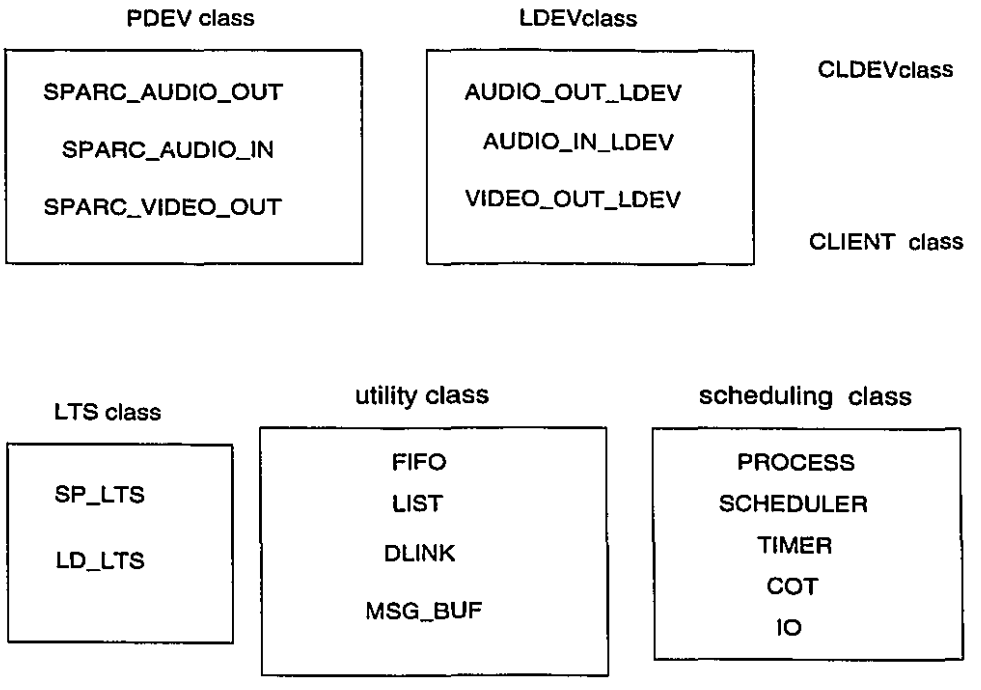

Figure 2.3: ACME Classes

4. arrange the classes into hierarchies

The process is iterative, working at both the high and low levels of abstractions at all stages. The class descriptions are successively refined throughout this design process. The goals of the object oriented approach are to make the writing of large program simpler, easier to change and maintain.

Corresponding to the abstractions mentioned in section 2.2, the ACME server program is structured with the following classes: PDEVs, LDEVs, CLDEVs, CLIENTs, LTS, utility and scheduling, as shown in Figure 2.3. Listed within each box, Figure 2.3 also shows the classes derived from each class.

The ACME's PDEV class attributes define the lists of strand types supported, the LDEVs attached, the LTSs controlled by the PDEV, the interrupt period of PDEV, an error tolerance for stream synchronization, the device data transfer direction (input or output), the medium (audio or video), and the device physical location.

In order to receive an audio/video file, the LDEV is attached to the PDEV and

a FIFO buffer is allocated. The audio output process receives 8-bit μ-law data from the network message buffer, converts it to the word-aligned 12-bit linear data, and scales it by volume in the FIFO. For video files, uncompressed video output data is received. After finishing a block of data input or data output, the ACME will update the CLDEV's logical time system. The LTS mechanism is used to synchronize the data flow to maintain the desired alignment of the audio/video outputs.

Remote procedure calls (RPC) and Continuous Media (CM) connections are used to transfer audio/video data between ACME servers. ACME can handle multiple RPC and CM connections for the concurrent transfer of multiple data streams on a workstation.

ACME handles error or failure conditions during its network data transfer by generating error messages and managing the recovering of resources.

Chapter 3 A Virtual Laboratory Design

A virtual laboratory is proposed with the motivation of providing a computerized tutorial system in a virtual world environment for education, training and sale presentations.

In the virtual laboratory design, several client's workstations are able to access the server's file system that is used to store the files associated with the tutorial demonstrations.

Part of the virtual laboratory environment includes sounds and video sequences which can be presented to the clients in the demonstrations. The clients can also send voice commands from the local workstation to the voice recognition toolkit in the server workstation via a TCP/IP connection with the audio/video server for hands-free interactive presentations. Using the audio/video server, the server and the clients can conduct real-time voice communication on TCP/IP connections.

In this chapter, we will introduce the design of a virtual laboratory.

3.1 Virtual Laboratory Design Elements

In order to design a virtual laboratory, we must first understand the concept of a virtual world. The virtual world is composed of the following elements :

- Object: unconnected three dimensional shapes that can be interacted with in-

dependently .

- Backdrop: basic structure that the virtual environment is built on.
- Viewpoints: the visible scene of the viewer in the virtual world. The viewer can control the course of the virtual world by controlling the viewpoint with a sensor, such as a joystick.
- Sensor: physical devices that respond to a change in physical conditions, such as head-tracking sensors and data gloves sensors that are used in a virtual world.
- Lights: multiple sources of light can be located anywhere in the world.
- Universe: the current simulation environment that contains all the relevant elements. It includes objects and related entities, such as lights, sensors, viewpoints, and sound channels.
- Simulation management: composing computational resources, interface devices and system resources to generate a simulated reality.
- Interface to the simulation manager: making decisions on the user's accessibility to the tool and its flexibility.
- Device support: being able to use various interface devices.
- Object control: interacting with an object and modifying its attributes, such as shape, location and appearance.
- Simulation control: A simulation may be controlled by various attributes.
- Modeling support: to read common file formats and for interactive creation of objects within the virtual environment.
- Compiled libraries: application programming interface (API) or toolkit. the tool for developing complex, stand-alone virtual reality applications to create a virtual environment.

- Scripted language: simplifies the creation of simulations. It usually uses an interpretive environment where changes are instantly incorporated into the simulation. Scripting generally trades off some flexibility for a much simpler interface to the simulation manager.
- Graphical: a tool for the point-and-click crowd. Provides intuitive interface to the simulation manager.
- Event loop: The computer runs an application by repeatedly looping through the event sequence until the simulation quits.
- Query sensors: Setting each input sensor attached to the reality engine to its current state.
- Update objects: The sensors use the stored information to update the position and orientation of any attached objects
- User actions: determine the interactions between the user and the virtual environment.
- Object tasks: any object can have a task that is executed automatically.
- Render: The virtual world will be rendered (drawn) periodically to respond to motion changes and user interactions.
- Geometry: A collection of a series of 3-D coordinate values that describes 3-D structures.

With the above virtual world design elements in mind, we will now present the functional description of a virtual laboratory [Pimentel and Teixeira, 1993b].

3.2 Virtual Laboratory Functional Description

The virtual laboratory design is intended to offer a simulated laboratory that would place the virtual instruments and virtual tools in the computer generated virtual environment. In the virtual laboratory, the users can watch a laboratory demonstration and actually pick up a virtual tool, do virtual experiment and observe the experimental results from the virtual instruments.

Compared to the conventional laboratory, the virtual laboratory has the advantage of safe, easy maintenance, cost efficiency, resource sharing, accuracy and interaction with the experiments. This is because the virtual laboratory will not cause any damage to the user or the virtual instruments which might be the case for the hardware instruments. It is not necessary to purchase all kinds of hardware instruments. The virtual laboratory resources are stored in the file system that can be shared by all the users. The output results are computed by the software engine such that it will always generate accurate results according to the input data. The user can interactively operate the virtual instruments during the experiments.

In the virtual laboratory, the server's file system is used to store the virtual laboratory main program, the diagnosis program, such as an integrated fault diagnosis system and laboratory demonstrations based on simulations using SPICE or MATLAB computations. Via the TCP/IP connection, the client's workstations are able to access the server's file system and perform the virtual laboratory experiments with sound and video sequences. Figure 3.1 presents the TCP/IP inter connection of the server workstation and the client workstations in the Virtual Laboratory. The server workstation and client workstations perform real time audio voice communications among each other by using the ACME server. The ACME server can be used in the virtual laboratory environment to create a server for the continuous media (video/audio) applications. For our experimental work, we have used SUN SPARCstations linked in a TCP/IP network.

The users of the virtual laboratory can also send voice command from the local workstation to the computer voice recognition toolkit in the server workstation via

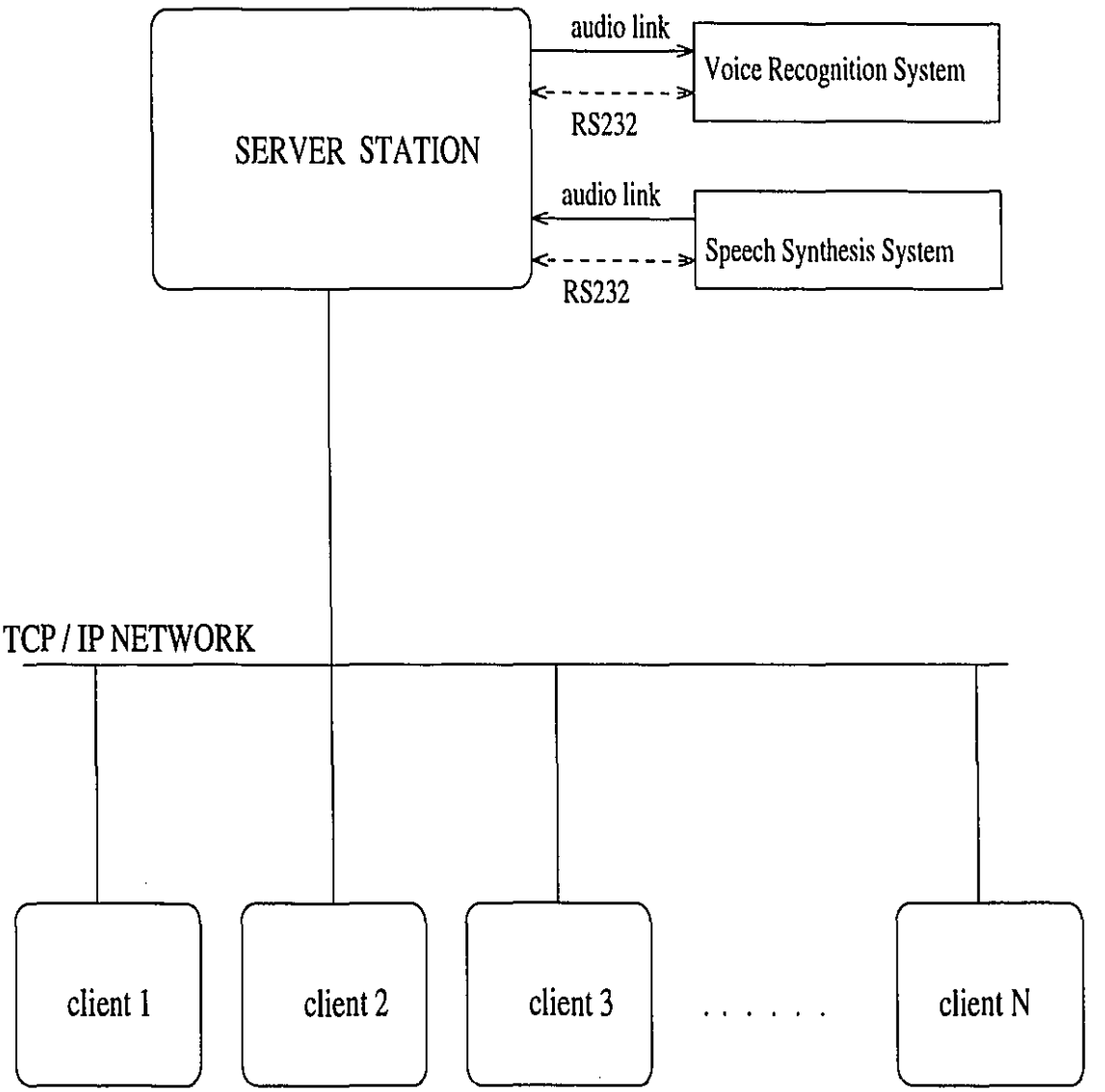

Figure 3.1: Virtual Laboratory Connections.

the TCP/IP connection with the audio/video server for hands-free interaction. A computerized speech synthesis system can also complement the audio presentation capabilities.

The Verbex Series 6000 Conversational Voice I/O system is an example of a speech recognition hardware system that connects to a workstation on a RS232 serial line. The Verbex Voice Conversational I/O System is a computer peripheral which translates a voice signal from a headset microphone into data and sends the data to the computer. The host computer can then send data to the recognizer's built-in speech

synthesizer to convert it into a speech output signal fed to the recognizer's headset earphones, or to the workstation's audio input jack for subsequent re-transmission to a client workstation. Similar audio systems exist from Covox or Creative Labs. Such products are PC based systems which can be linked to the same workstation by RS232 serial lines. Bus oriented peripheral systems are also available for the SUN workstations but at a significantly higher cost.

With this proposed virtual laboratory environment, many novel applications can be developed. For example, the virtual laboratory main program could call on an integrated fault diagnosis system to find faults in digital circuits. Such a system could solve troubleshooting problems and offers explanations during the process of troubleshooting a circuit within an intelligent tutoring system [Sandrasegaran, 1993]. It is envisaged to run the Integrated Fault Diagnosis System for Physical Systems in the virtual laboratory environment, where the sound, audio and video can be integrated to it.

Another example of a laboratory demonstration program that can be called from the virtual laboratory main program would invoke a SPICE simulation. The students in a virtual laboratory may put on data gloves and a helmet mounted display and use the data gloves or voice to interact with the SPICE simulation. They may pick up 3-D component objects and build a 3-D circuit with one hand, start a simulation with voice commands, and the computer will explain all the simulation process, show the faulty portion of the design and give instructions with sound, text, images, and video sequences. The students could also interactively ask questions and get answers from the computer. The user may pick up an oscilloscope object, attach its virtual probes to the virtual circuit and observe the signal waveform of the circuit at that point.

Chapter 4 Implementation and Experimental Results

In order to assess the suitability of the ACME audio/video server for the proposed virtual laboratory environment, an interactive test program was developed to exercise audio/video interactions in this client/server model. This program was implemented using the ACME server, X11 library, and the InterViews interface builder, IBUILD. Since we are using InterViews version 3.1, it was necessary to upgrade the ACME code from its previous implementation using version 3.0.1. Appendix A presents the necessary changes.

4.1 The Audio/Video Server Test Program

Figure 4.1 presents the user interface window of the audio/video server test program which runs on SUN workstations. The user interface window contains seven buttons: Local Video, Local Audio, Record Audio, Send Video, Send Audio, Clear, and Exit. The users can press one or more pushbuttons to playback a sound or a video sequence, to record a voice message, and to send voice messages or video sequences to the server or client workstation for interactive demonstration. The pushbutton operations are as follows:

- The Local Video button starts the process of playing pre-recorded video sequences on the local workstation.
- The Local Audio button starts the process of playing a pre-recorded audio file

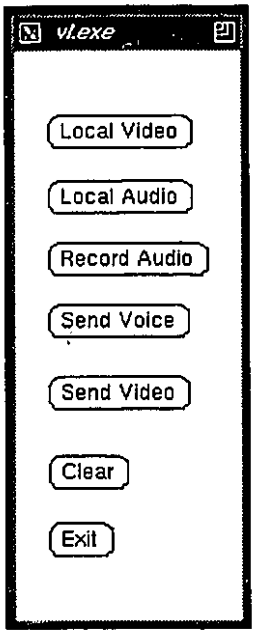

Figure 4.1: User Interface Window.

on the local workstation. The file being played could have been generated by the speech or audio synthesis facility.

- The Record Audio button starts the process of recording an audio file from a SUN SPARCstation's microphone on the local workstation.
- The Send Voice button starts the process of sending a pre-recorded voice message file to another workstation which could provide the voice recognition facility.
- The Send Video button starts the process of sending a pre-recorded video sequence from the file server to another workstation.

- The Clear button starts the process of clearing the video window on the local workstation.
- The Exit button starts the process of killing all the processes that have been started by operating the pushbuttons in the user interface window on a workstation.

Since the voice recognition hardware was not available during this research, it could not be tested and it remains to be implemented in a future project. For these tests that were carried out, the SUN SPARCstation microphone was attached to the audio input jack of the client workstation. The built-in SUN SPARCstation's audio speaker was used for the voice output device.

The software programs including the ACME and InterViews libraries are stored in the server's file system that can be accessed by each of the client workstations.

4.2 Software Architecture of the Audio/Video Server Test Program

InterViews libraries are used in the audio/video server test program for implementation of our window-based user interface, as well as for supporting the operation of ACME's functions. InterViews is object-oriented and its active elements, such as windows, buttons, menus and documents, have inherited behavior. With InterViews, an object of a user interface gives an active view of some data. InterViews uses a set of classes to define the behavior of user interface objects. It can be characterized into two categories: protocols and kits. A protocol defines an operation set of an object, such as drawing or handling input. A kit defines an operation set for creating other objects and can be considered as an object factory. Using kits can avoid showing the details of object generation and the tradeoffs between subclass and instance during implementation. In this way, a higher-level organi-

zational structure can be provided to the system. InterViews can be compiled with any C++ Version 2.0, 2.1, or 3.0 compatible compiler on X11R4 or X11R5. [Linton et. al., 1992]

The InterViews interface builder program IBUILD is used to develop the graphical interface of pushbuttons described in section 4.1 and to automatically generate the corresponding C++ codes. By simply modifying the IBUILD generated C++ code, we can easily implement the pushbutton functions for running the final test program..

Figure 4.2 presents the IBUILD utility window and each of the top-level user interface pushbuttons assembled or constructed using the IBUILD utility window. This user interface contains pushbuttons for Local Video, Local Audio, Record Audio, Send Video, Send Audio, Clear, and Exit. With the IBUILD utility window, the user interface main program design can be generated by setting instances for each class such as pushbuttons, menu items, dialog boxes for the disired test program. With the IBUILD utility window, C++ classes called core class and core subclasses can also be generated. From the IBUILD utility window, the IBUILD Pushbutton Class Building Window shown in Figure 4.3 can be obtained and used to set the class name, member name, button state's name, and button state's setting value for the pushbutton class. From the IBUILD Pushbutton Class Building Window, the IBUILD Button State Setup Window shown in Figure 4.4 can then be obtained to set the initial value and member function name using the proper entry fields in the dialog box.

The core class vl contains all the variables of the exported members in the user interface composition. The exported button state svidBS (see Figure 4.3) becomes a protected member of the core class that is not accessible to other objects. A member function svided specified in the interface (see Figure 4.4) correspondingly has public virtual member functions in the core class, and it does nothing by default. The virtual functions of the core class that operate in the test program can be rede-

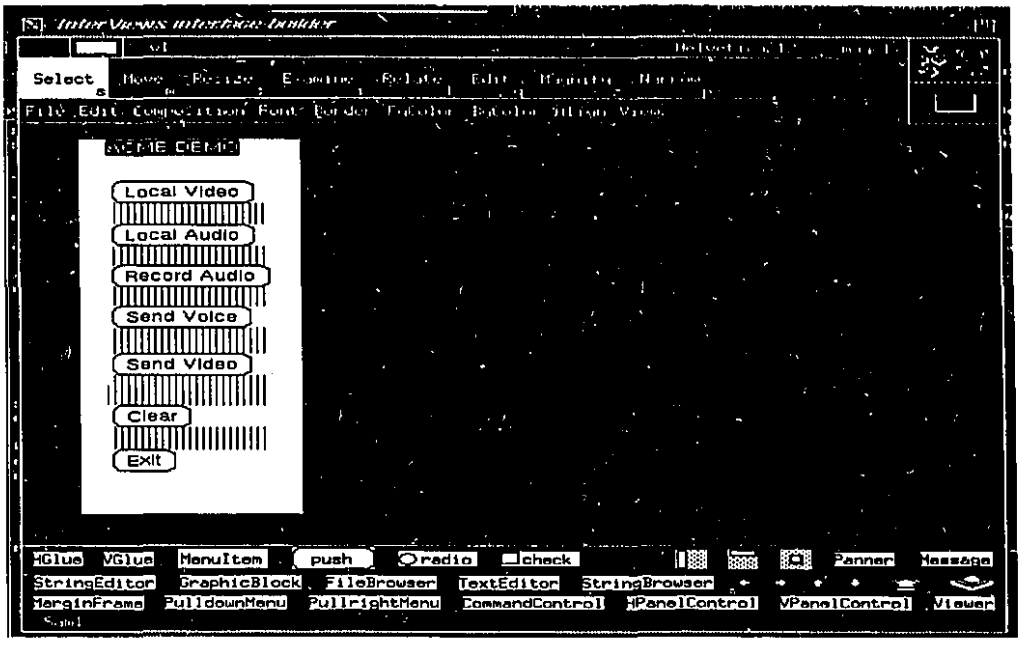

Figure 4.2: IBUILD Utility Window

fined. Later, if the appearance of the user interface needs to be modified, we need only regenerate the files of the core class that was not modified. Therefore it will not affect the changes of the core subclass. Then the test program application can be recompiled to reflect the modifications we made for the user interface.

The core subclass can be modified to interface with the rest of the test program application. The member functions can be defined to provide controlled access to the variables of the exported member accessible by other class members. Appendix B gives the examples of the virtual member function implementations used in the test program. In order to enable the functions of pushbuttons in the user interface, we need to modify the core subclass. During core subclass modification for the user interface, we can reset the value of the button state to "0" to un-highlight the push-button, or set the button state's value to equal the setting value to highlight the pushbuttons.

We can also change the appearance of the user interface in the Ibuild utility window and regenerate the prototype of the test program user interface without affect-

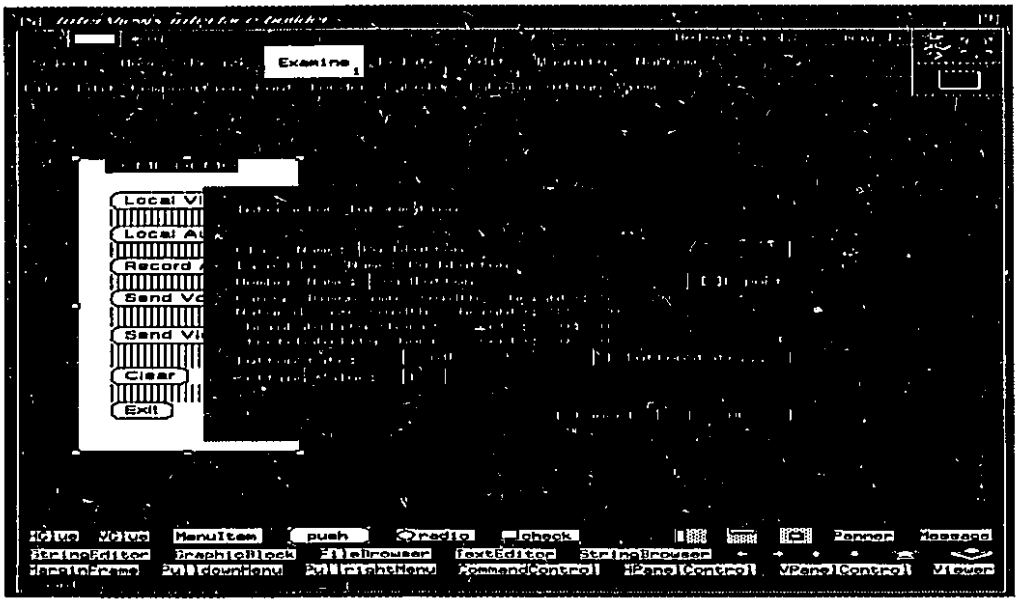

Figure 4.3: IBUILD Pushbutton Class Building Window

ing the changes of the core subclass. We can use the "Generate..." command in the File menu of the Ibuild utility window to generate a modified user interface. We can choose to overwrite all the generated files that already exist except the core subclass files by default. Users can also make their own choices. [Vlissides, 1991]

The IBUILD generate function produces the C++ code that implements the user interface. For each MonoScene subclass contained in the user interface, IBUILD will correspondingly generate a set of four files for each MonoScene subclass, where each file name is prefixed by the subclass name. Figure 4.5 presents the IBUILD MonoScene Building Window. With the MonoScene Building Window, we can set the class name for the top level user interface which will be used in starting up the application program.

4.3 Implementation of the Audio/Video Server Test Program

The server's file system is used to store the main test program and its subroutines, including any associated sound or video sequences. Via the TCP/IP con-

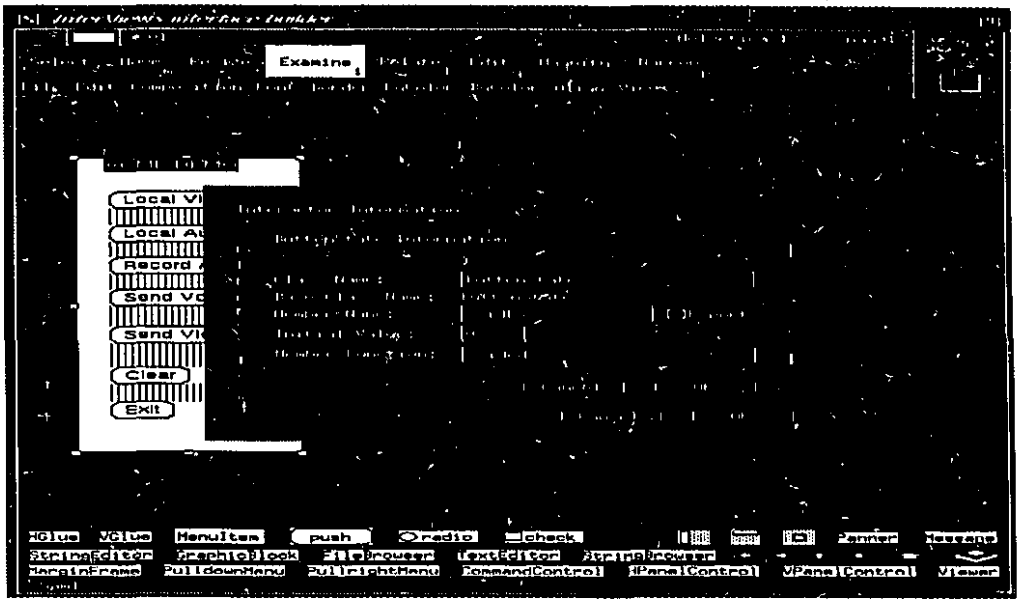

Figure 4.4: IBUILD Button State Setup Window

nection, the workstations in the network are able to access the file server and perform the required multimedia demonstrations involving the playback of sound and video sequences.

Using this test program, the workstations in the network are also able to send audio or video sequences to each other via the ACME network server.

Figure 4.6, 4.7 and 4.8 present sample results using the test program video diplay window where three video frames were captured from the screen video sequence. The user interface window concurrently displays the buttons for Local Video, Local Audio, Record Audio, Send Video, Send Audio, Clear, and Exit, and multiple selections are possible.

In the modification of the core subclasses of the IBUILD generated C++ code, the *system* call is used to connect the pushbuttons in the user interface window to the command actions:

- playback video sequence on the local workstation
- playback audio on the local workstation

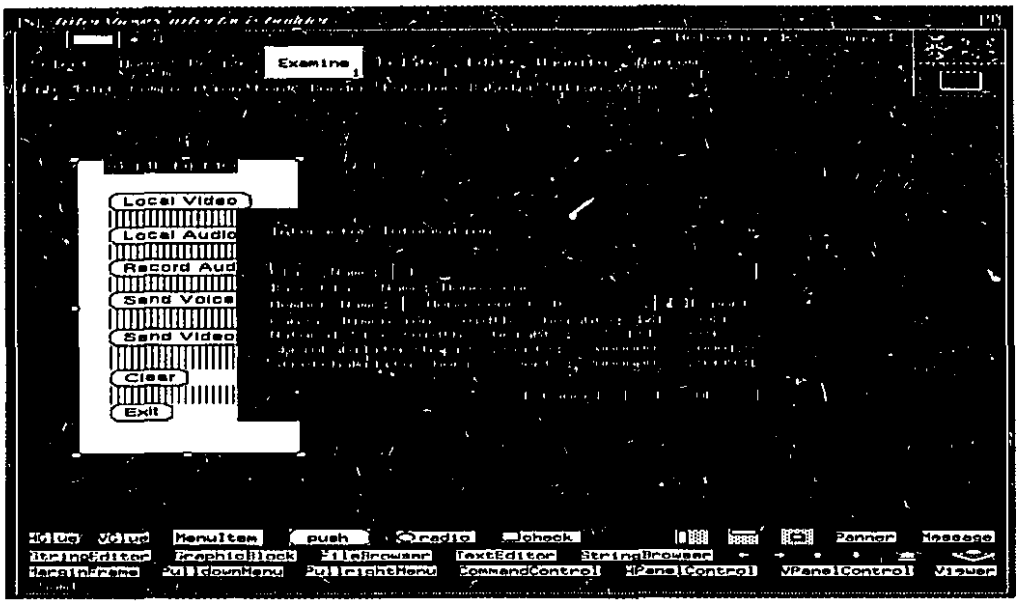

Figure 4.5: IBUILD MonoScene Building Window

- record audio on the local workstation
- send audio voice to the remote workstation
- send video to the remote workstation
- clear the display window
- exit the test program

Before the *system* call to execute the corresponding virtual member function, the "fork" function is used to create a new process. In this way, multiple processes are started by repeated pushbuttons selections in the user interface window.

The *system* call issues a shell command in the form *system(string)*. The "system" function gives the string to the shell as input, just as if the string had been typed as a command from a terminal.

For example, the *system* call for the Local Video button function is implemented as *system("xterm -fg white -bg black -exec src/apps/studio/studio src/apps/studio/script.playback")*. It creates a window where the function studio using the parameter information from the file *script.playback* plays back a video sequence in the window of the local workstation.

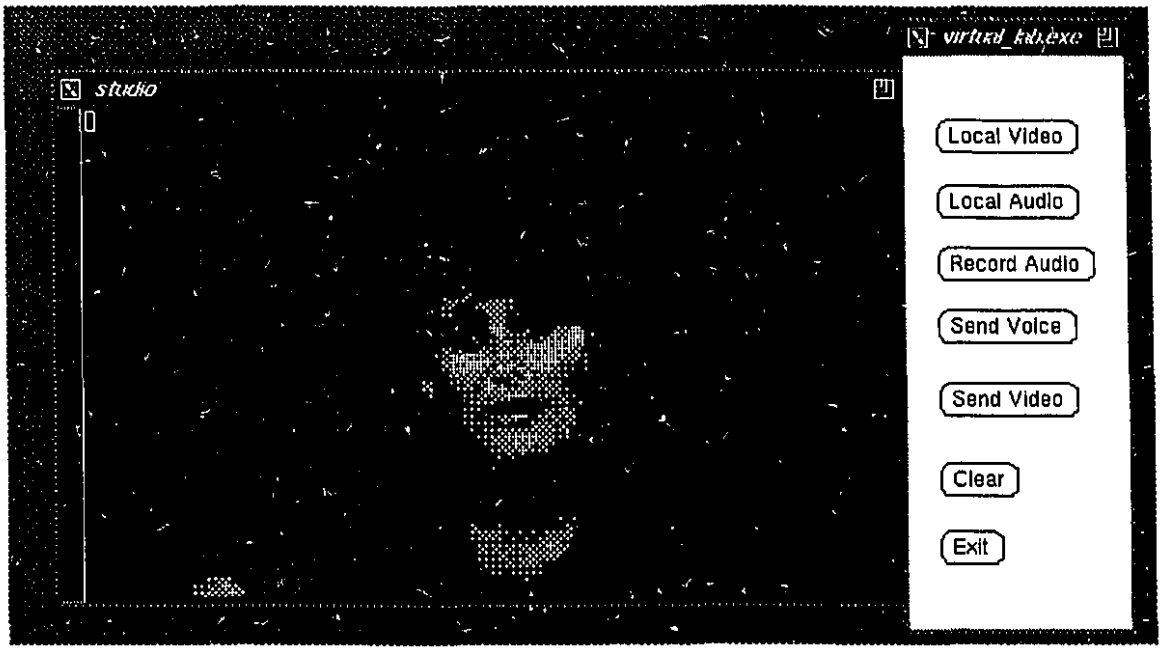

Figure 4.6: Sample Diplay Window - Video Frame I

Similarly, the *system* calls for the Local Audio, Record Audio functions are implemented as *system("studio script.pa")* and *system("studio script.r")* to play back or record audio on the local worksation.

The *system* call for the Send Voice function is implemented as *system("studio script.pa")*. It plays an audio message from the local workstation to a remote workstation. This is done by setting the ACME environment for the remote workstation (haley) which is implemented as *putenv("ACME=haley")* .

Similarly, the *system* call for the Send Video function is implemented as *system(" studio script.playback")* after setting the ACME environment to the remote worksta-

Figure 4.7: Sample Diplay Window - Video Frame II

tion. It plays a video sequence from the file server disk to a remote workstation.

The *system* call for the Clear function is implemented as *system ("clear")*. It clears the video display window.

The *system* call for the Exit function is implemented as, *system(s)*, where "s" is a string generated from *sprintf(s, "kill %d %d %d %d %d",pid1, pid2, pid3, pid4, pid5)* . The "kill" function is used to send the terminate signal to the specified process IDs, pid1, pid2, pid3, pid4 and pid5. The specified process IDs were determined in the return codes from the "fork" calls. The "kill" does not send a signal to the current job.

Figure 4.8: Sample Diplay Window - Video Frame III

Appendix B gives the source code of the core subclasses implementation for the buttons Send Video, Clear, and Exit.

To start the application, we use the command: *studio script-command-file.*

The studio program accesses parameters contained the script command file in the following formats:

. *audio_out audio_filename start_time end_time vol*
. *audio_in audio_filename start_time end_time*
. *video_out video_filename start_time end_time x0 y0 xs ys*

where all times are expressed in milliseconds from the beginning of the file.

For our implementation, the studio program provided in the ACME package was also recompiled for InterViews version 3.1.

The size of the upgraded ACME server executable file, acme, is 90112 Bytes while its original size was 114688 bytes. The size of the test program executable file, vl.exe, is 1384448 Bytes. The size of the upgraded video/audio application executable file, studio, is 196608 Bytes compared to its original size of 507904 Bytes.

4.4 Experimental results

With the ACME environment previously set up in the .cshrc, the ACME server is started by typing *acme* in one window and the user interface test program is started by typing *vl.exe* in another window.

The following responses were obtained for Test RUN I by exercising the test program:

- Pressing the Local Video button to playback video sequence on the local workstation.

  It took 2 sec. of latency to have video images appear on the screen without any window on the local workstation, and 5 sec. of latency to have video images appear on the screen when a *system* call to create an X-window was added before the *system* call of playback a video sequence in the pushbutton function implementation.

  For video sequence file sizes of 1900800 Bytes and 950450 Bytes, the video sequence playback lasted 24 sec. and 9 sec. respectively.

- Pressing the Local Audio button to playback audio on the local workstation:

  It takes 2 sec. for the audio to be heard on the local workstation.

For audio file of 370712 Bytes, 423960 Bytes, 452632 Bytes, and 1212440 Bytes, the audio sequence playback lasted 46 sec., 52 sec., 60 sec., and 150 sec. respectively.

- Pressing the Record Audio to record audio from the local workstation,
  The recording starts in 1 sec.
  The size of the recorded file can be pre-set from the audio_in command in the script file script.pa.

- Pressing the Send Voice button to send an audio voice file to a remote workstation, it took 5 sec. for the user to hear the audio on the remote workstation. For audio files of 370712 Bytes, 423960 Bytes, 452632 Bytes, and 1212440 Bytes, the audio sequence playback lasted 46 sec., 52 sec., 60 sec., and 150 sec. respectively.

- Pressing Send Video button to send video to a remote workstation, it took 8 sec. to have a video image appear on the screen of the remote workstation. For the video sequence file sizes of 1900800 Bytes and 950450 Bytes, the video sequence playback lasted 24 sec. and 9 sec. respectively.

- Pressing the Clear button to clear the display window,
  it takes 2 sec. to clear the diaplay window.

- Pressing the Exit button to exit the test program required 2 sec. to exit.

These operations were repeated in Test RUNs II and III and the results of the three tests are given in Tables 4.1, 4.2, and 4.3.

Pushbuttons	File Size (Bytes)	Delay (Seconds)	Duration (Seconds)
Local Video	1900800	2	24
	950400	2	9
Local Audio	370712	2	46
	423960	2	52
	452632	2	60
	1212440	2	150
Record Audio		1	
Send Voice	370712	2	46
	423960	2	52
	452632	2	60
	1212440	2	150
Send Video	1900800	8	24
	950400	8	9
Clear		2	
Exit		2	

Table 4.1: Experimental Results - Test Run I.

Pushbuttons	File Size (Bytes)	Delay (Seconds)	Duration (Seconds)
Local Video	1900800	1	23
	950400	2	10
Local Audio	370712	2	48
	423960	2	52
	452632	2	59
	1212440	2	148
Record Audio		1	
Send Voice	370712	2	46
	423960	2	51
	452632	2	60
	1212440	2	149
Send Video	1900800	8	24
	950400	7	10
Clear		1	
Exit		1.5	

Table 4.2: Experimental Results - Test Run II.

Pushbuttons	File Size (Bytes)	Delay (Seconds)	Duration (Seconds)
Local Video	1900800	2	22
	950400	2	9
Local Audio	370712	2	46
	423960	2	50
	452632	2	59
	1212440	2	149
Record Audio		1	
Send Voice	370712	2	47
	423960	2	52
	452632	2	59
	1212440	2	147
Send Video	1900800	7	24
	950400	7	10
Clear		2	
Exit		2	

Table 4.3: Experimental Results - Test Run III.

From these experimental results, we can derive relationships between the audio or video playback time and audio or video file size, the maximum number of clients that could be supported in the virtual laboratory, and the relationship between the overhead or the frame rate and the window size of the video display as follows:

- The relationship between audio or video playback time and audio or video file size:

  Figure 4.9 and Figure 4.10 show the audio/video performance in the virtual laboratory application. From Figure 4.9, we can derive the formula for The Relationship Between the Audio Playback Time and Audio File Size:

  The slope of the curve in the Figure 4.9 is,

  $slope = \frac{(124-100)}{(10-8)\times 10^5} = 12 \times 10^{-5}(seconds/byte)$.

  The audio playback time will be:

  $TIME = slope \times AUDIO\_FILESIZE = 12 \times 10^{-5} \times AUDIO\_FILESIZE$

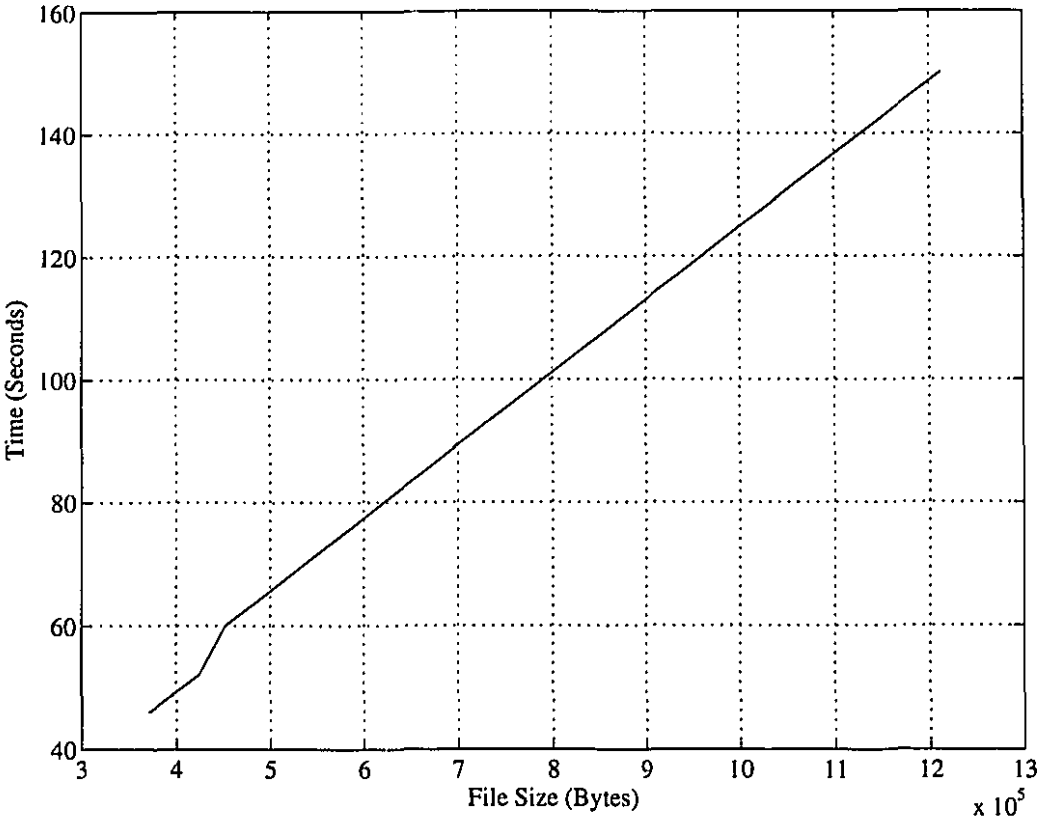

Figure 4.9: Relationship Between Audio Playback Time and Audio File Size

From Figure 4.10 we can derive the formula for The Relationship Between the Video Playback Time and Video File Size:

The slope of the curve in the Figure 4.10 is,

$slope = \frac{(19.2-16)}{(1.6-1.4)\times 10^5} = 1.6 * 10^{-5}(seconds/byte)$.

The video playback time will be,

$TIME = slope \times VIDEO\_FILESIZE = 1.6 \times 10^{-5} \times VIDEO\_FILESIZE$

Obviously we would prefer a higher slope for the relationship between the playback time and the audio/video file size. That means we would prefer a smaller video/audio file size for the same duration of the video/audio play-

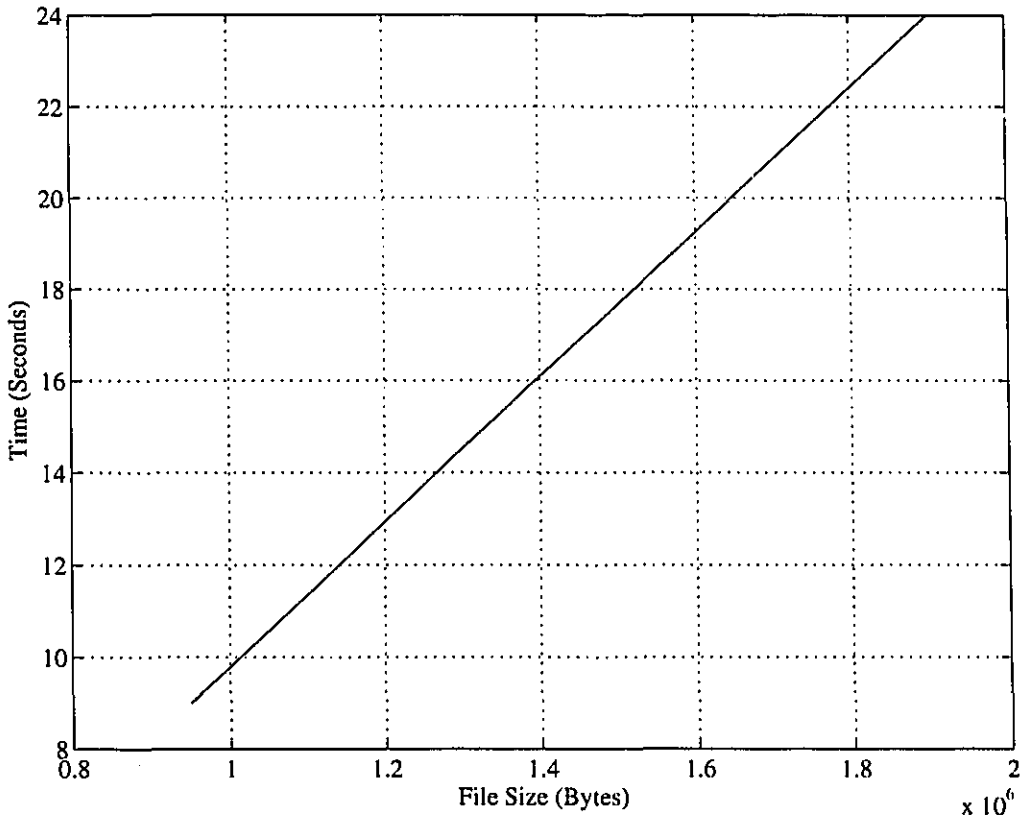

Figure 4.10: Relationship Between Video Playback Time and Video File Size

back. This higher slope can be achieved by employing video/audio compression and decompression technology in the future development.

- The maximum number of clients that could be supported in the virtual laboratory is estimated as follows:

  Theoretically, an 8KHz sampling rate with 8 bits per audio sample gives a data rate of:

  $8bits/sample \times (8 \times 10^3)samples/sec. = 64(kbps)$

  For monochrome video using 1 bit per pixel at 15 frames per second in a 352 by 288 display window, the data rate is:

$1bit/pixel \times 15frames/sec. \times (352 \times 288)pixels/frame = 1520.6(kbps)$

For monochrome video using 1 bit per pixel at 15 frames per second in a 320 by 240 display window, the data rate is:

$1bit/pixel \times 15frames/sec \times (320 \times 240)pixels/frame = 1152(kbps)$

Since the maximum data rate on the TCP/IP connection is 10 Mbps, we can estimate the maximum number of clients that can be supported simultaneously in our virtual laboratory as follows:

For monochrome video using 1 bit per pixel at 15 frames per second in a 352 by 288 display window and 8KHz sample rate with 8 bits per sample audio, the maximum number of clients that can be simultaneously supported in our virtual laboratory is:

$\frac{10000kbps}{1520.6kbps+64kbps} = 6(clients)$

For monochrome video using 1 bit per pixel at 15 frames per second in a 320 by 240 display window and an 8KHz sampling rate with 8 bits per audio sample, the maximum number of such clients that can be simultaneously supported in our virtual laboratory is:

$\frac{10000kbps}{1152kbps+64kbps} = 8(clients)$

- The relationship between the networking overhead or the frame rate and the window size of the video display:

  From the experimental results in Figure 4.9, we have:

  $slope = 12 \times 10^{-5}(seconds/byte)$

  So the data rate of the audio is:

  $8bits/byte \times \frac{1}{12\times10^{-5}}bytes/sec. = 66.66(kbps)$

  From the experimental results in Tables 4.1, we derived the data rate for monochrome video using 1 bit per pixel at 15 frames per second in a 352 by 288 display window as follows:

  $8bits/byte \times \frac{1900800}{24}bytes/sec. = 633.6(kbps)$

From the experimental results in Tables 4.1, we derived the data rate for monochrome video using 1 bit per pixel at 15 frames per second in a 320 by 240 display window as follows:

$8bits/byte \times \frac{950450}{9} bytes/sec. = 844.8(kbps)$

Comparing the above experimental test results with the previous theoretical results, we observed that:

The overhead for monochrome video using 1 bit per pixel at 15 frames per second in a 352 by 288 display window is:

$1520.6kpbs - 633.6kpbs = 887kpbs$

and this corresponds to the actual reduced video frame display rate of

$\frac{(633.6\times10^3)bits/sec.}{1bit/pixel\times(352\times288)pixels/frame} = 6(frames/sec.)$

The overhead for monochrome video using 1 bit per pixel at 15 frames per second in a 320 by 240 display window is:

$1152kpbs - 844.8kpbs = 307.2kbps$

and this corresponds to the actual reduced video frame display rate of

$\frac{(844.844\times10^3)bits/sec.}{1bit/pixel\times(320\times240)pixels/frame} = 11(frames/sec.)$

From the above results, we see that larger video display windows result in a lower data rate, more overhead, and a lower frame rate. Smaller video displays are recommended.

4.5 Discussion and Future Development

These experimental results are encouraging for using an audio/video server in the virtual laboratory context. First we recognize that the maximum client workstation count is optimistic since the efficiency of Ethernet is known to fall off under heavy loading. But there are a number of strategies which can greatly improve

the situation. The first would involve using video compression on the files together with video hardware for decompression and display of the video sequences at each client station. The technology exists which can deliver TV quality color images using 100KBytes per second. With the current interest in multimedia systems, the price and performance of video compression hardware is expected to improve significantly. A second factor is that the video sequence should not be required continuously at the client workstation. The virtual laboratory context also requires time for the user to analyze a situation, make decisions, and operate the virtual equipment. Finally, by configuring a local network for the virtual laboratory only, unwanted network loading can be avoided.

Eventually the virtual laboratory is envisaged exploiting the concepts of virtual reality (VR). But many of its technologies such as helmet mounted displays or datagloves are still in their infancy. As VR technology evolves and the prices drop, we expect to integrate more of these into the virtual laboratory implementation offering laboratories with virtual tools and placing the laboratory in a virtual reality environment.

For future development of the virtual laboratory the following areas are suggested:

- ACME uses a special format for video files and this should be extended to handle popular standard formats such as MPEG.
- Using compression/decompression technology to reduce the file size of the video sequence. The quality of the video sequence can be improved by improving the video frames speed to 30 frames per second. [Miyamoto et. al., 1992]
- Combining the TCP/IP network with the Asynchronous Transfer Mode (ATM) switching network for the virtual laboratory demonstration could increase the network bandwidth and hence improve the quality of the video playback

[Tanabe et. al., 1991], [Kato et. al., 1990], [Kuwahara et. al., 1989], [Endo et. al., 1990], [Gutfreund et al., 1992], and [Sakurai et. al., 1990].

- Adding a gesture interface (the Dataglove) to manipulate objects in the virtual laboratory.
- Implementing the voice recognition/synthesis facility in the server workstation to support voice command operation on the client workstations.
- Integration of different virtual reality software toolkits such as WorldToolkit by Sense 8 for convenient application development.

The future virtual laboratory development is very promising, some work is currently underway and we expect to develop a virtual laboratory prototype incorporating 3D computer generated images, audio sounds, and computer voice interface in the near future.

Chapter 5 Conclusion

This thesis began with an overview of state-of-the-art technologies in the areas of virtual reality systems and multimedia applications. The design of a virtual laboratory framework based on the ACME audio/video server was outlined. The ACME video/audio network server was upgraded to run with the current version of InterViews (V3.1). A library of subroutines was developed to support the proposed functions within this architecture. A test program was developed to evaluate the performance of the audio/video file serving.

Using its user interface window, a user can press pushbuttons to playback a sound record or a video sequence, to record a voice message, and to send a voice message or a video sequence to the server's workstation for interactive operation of one or multiple selections. The server and client workstations were demonstrated to perform real time audio and video communications between each other via the continuous media (Video/Audio) server on TCP/IP connections. The clients can also send voice command from the local workstation to the server workstation for processing by a voice recognition toolkit for hands-free interactive presentations. The system performance was tested and the results were evaluated.

Finally future improvements for the development of a virtual laboratory were discussed. The study has explored the possibility of adapting the TCP/IP network for audio/video presentations, voice communications, and hands-free interactive demonstrations in a virtual laboratory.

Chapter 6 REFERENCES

[Addeo and Weinstein, 1992] E.J. Addeo and S.B. Weinstein, "VideoWindow: Experimentation With A More Natural Form of Audio/Video Teleconferencing", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 7-9.

[Akeley, 1991] Kurt Akeley, The Silicon Graphics 4D/240GTX Superworkstation, IEEE Computer Graphics and Applications, pp. 71, Jul., 1989, pp. 71.

[Anderson et al., 90a] D. P. Anderson, S.Y. Tzou S.Y., R. Wahbe, and M. Andrews, " Support for Continuous Media in the DASH System". Proc. 10th International Conference on Distributed Computing Systems, Paris, May 1990.

[Anderson et al., 90b] D. P. Anderson, G. Homsy, R. Govindan, "Implementation Issues for a Network Audio/Video Server", Technical Report UCB/CSD 90/597, University of California at Berkeley, Sept., 1990.

[Anderson, 90c] D. P. Anderson, " Meta-Scheduling for Distributed Continuous Media", Technical Report UCB/CSD 90/599, University of California at Berkeley, Oct., 1990.

[Anderson, 91a] D. P. Anderson, R. G. Herrtwich, and C. Schaefer, "SRP: A Resource Reservation Protocol for Guaranteed Performance Communication in the Internet" Internal Report, University of California at Berkeley, 1991.

[Anderson et al., 91b] D. P. Anderson, G. Homsy, R. Govindan,"Abstractions for Continuous Media in a Network Window System". International Conference on Multimedia Information Systems, Singapore, Jan. 1991.

[Anderson, 92] D. P. Anderson, Y. Osawa, R. Govindan, " A File System for Continuous Media", ACM Transactions on Computer Systems, Vol. 10 No. 4, Nov. 1992, pp. 53-54.

[Bates et al., 1990] P.C. Bates, and M.E. Segal, "Touring Machine: A Video Telecommunication Software Testbed", Proceedings First International Workshop on Network and Operating System Support for Digital Audio and Video, Berkeley, CA, Nov., 1990.

[Bolt, 1985] R.A. Bolt, "Conversing with Computers "Technology Review, Feb., 1985, pp. 34-43.

[Bohm et al., 1992] K.W. Bohm and K. Vaanaen, "Given: Gesture Driven Interactions in Virtual Enviroment. A Toolkit Approach to 3D Interactions". Proceedings

Interfaces to Real and Virtual Worlds, Montpellier, France, Mar.,1992, pp. 243-254.

[Bricken, 1990] W. Brichen, "Extended Abstract: A Formal Foundation for Cyberspace", Beyond The Vision, Proceedings of Virtual Reality '91, The Second Annual Conference on Virtual Reality, Artificial Reality, and Cyberspace, San Francisco, Sept. 23 - 25, 1991, pp. 9.

[Bricker, 91] D. C. Bricker, and R. Van Liere, "Multimedia Synchronization and UNIX", Proc. Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Springer Verlag, Nov. 18-19, 1991.

[Bryson and Gerald-Yamanski, 1992] S. Bryson, and M. Gerald-Yamanski, "The Distributed Virtual Windtunnel, Supercomputing '92 Proceedings, IEEE Computer Society Press 1992.

[Burdea and Lanangrana, 1992] G. Burdea and N. Langrana,"Virtual Force Feedback:lessons, challenges, future applications", Proceedings of 1992 A.S.M.E. Winter Annual Meeting - Advances in Robotics, DSC-Vol.,42, Anaheim, CA, Nov.,1992, pp. 41-47.

[Burdea and Zhuang, 1992] G. Burdea and J. Zhuang,"Actuator System for Providing Force Feedback to a Dextrous Master Glove", US Patent 5,143,505, Sept, 1, 1992.

[Casner et al., 1990] S. Casner, K. Seo, W. Edmond, and C. Topolcic, "N-Way Conferencing with Packet Video", Proceedings 3rd International Workshop on Packet Video, VISICOM '90, Morristown, NJ, Mar.,1990.

[Chapin and Foster, 1992] W. Chapin and S. Foster, "Virtual Environment Display for a 3D Audio Room Simulation", Proceedings of SPIE Stereoscopic Display and Applications, pp. 12, 1992

[Chao et al., 1989] J.Y. Chao, W.T. Lee, and L.Y. Kung,"A New Buffer Insertion Ring with Time Variant Priority Scheme to Facilitate Real-Time Image Transmission on a High Speed Integrated Local Area Network", IEEE 14th Local Computer Network, Minneapolis Minn., Oct. 1989, pp. 212-218.

[Clark et al., 1992] D.D. Clark, S. Shenker, and L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", Computer Communication Review, Vol.22, No.4, Oct, 1992, pp. 14-26.

[Cole, 1993] B. Cole, "The Technology Framework", IEEE Spectrum, March, 1993, pp. 36-37.

[Cordes et al., 1992] R. Cordes, H. Peyn, and T. Kummerow, "Access Methods for Distrubuted Multimedia Information Systems Based on Provite Broadband Communication Networks",Computer Communication Review, Vol.22, No.3, July, 1992, pp. 75-78.

[Coulson, 92a] G. Coulson, G. S. Blair, N. Davies, and N. Williams, "Extensions to ANSA for Multimedia Computing", Computer Networks and ISDN Systems, 1992, pp. 305-323.

[Coulson, 92b] G. Coulson, G. S. Blair, F. Hore, L. Hazard, and J. B. Stefani, "Supporting the Real-time Requirements of Continuous Media in Open Distributed Systems", International Standards Organization document ISO/IEC JTC1/SC21 IST21/-/1/5:94, 1993.

[Cutt, 1992] P. Cutt,"Tactools Technical Summary", Xtensory Inc, Scotts Valley, CA.

[Demers et al., 1989] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", Proceedings ACM SIGCOMM'89, pp. 3-12.

[Digiano and Buxton, 1993] Christopher J. Digiano and William A.S. Buxton, "Using Non-Speech Audio at the Interface", Virtual Reality System, Vol. 1. No. 1, Mar., 1993, pp. 60-62.

[Endo et al., 1990] N. Endo, T. Ohuchi, T. Kozaki, H. Kurahara, and M. Mori, "Traffic Characteristics Evaluation of a Shared Buffer ATM Switch", Proc. GLOBECOM'90, 1990, pp. 905.1.

[Grantham, 1991] Grantham, Charles E., "Visual Thinking in Organizational Analysis", Beyond The Vision, Proceedings of Virtual Reality '91, The Second Annual Conference on Virtual Reality, Artificial Reality, and Cyberspace, San Francisco, Sept. 23 - 25, 1991, pp. 70 -75.

[Gutfreund et al., 1991] V. Phuah, J. Diaz-Gonzalez, R. Sasnett, and S. Gutfreund,"Dynamicity Issues in Broadband Network Computing", Second International Workshop on Network Operating System Support for Digital audio and Video, Germany, Nov., 1991.

[Gutfreund, 1991] S. Gutfreund,"Integrating Distributed Visualization System with Multimedia", SIGGRAPH'91 Workshop on Computer Graphics in the Network Environment, July, 1991.

[Gutfreund et al., 1992] V. Phuah, J. Diaz-Gonzalez, R. Sasnett, and S. Gutfreund,"Developing Distributed Multimedia Applications", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 57-60.

[Hardy and Healy, 1993] Doug Hardy and Mike Healy, "Constructive and Virtual Interoperation: A technical Challenge", Virtual Reality System, Vol. 1. No. 2, Oct, 1993, pp. 12-16.

[Heinrichs, 1992] B. Heinrichs,"Versatile Protocol Processing for Multimedia Communications", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 49-50.

[Hyman et al., 1991] J. Hyman, , A. Lazar, and G. Pacific,"Real-Time Scheduling with Quality of Service Constrains", IEEE JSAC, Vol. 9, No. 9, Sept., 1991, pp. 1052-1063.

[Iwata, 1990] I. Iwata, "Artificial Reality with Force Feedback: Development of Desktop Virtual space with CompactMaster Manipulator", Computer Graphics, vol.24, No.4, 1990, pp. 165-170.

[Kernighan and Ritchie, 1978] B.W. Kernighan and D.M. Ritchie,"The C Programming Language", Prentice-Hall, 1978.

[Kuwahara et al., 1989] H. Kuwahara, N. Endo, M. Ogino, and T. Kozaki, "A Shared Buffer Memory Switch for an ATM Exchange", Proceedings ICC'89, 1989, pp. 118-122.

[Lang et al., 1988] J-J. Garcia-Luna-Aceves, E.J. Craighill, R. Lang, "An Open-System Model for Computer-Supported Collaboration", Proceedings 2nd IEEE Conference on Workstations, Mar., 1988.

[Latta, 1991] Latta, John N., "When Will Reality Meet the Marketplace?", Beyond The Vision, Proceedings of Virtual Reality '91, The Second Annual Conference on Virtual Reality, Artificial Reality, and Cyberspace, San Francisco, Sept. 23 - 25, 1991, pp. 109-141.

[Lazar, et al. 1990] A.A. Lazar, A. Temple, and R. Gidron, "An Architecture for Integrated Networks that Guarantees Quality of Service," Intl. J. of Digital and Analog Cabled Systems, Vol.3., No. 2, 1990.

[Lee and Kung, 1992] W.T. Lee, and L.Y. Kung, "Design A High-Speed Multimedia Network Based on More Reliable Priority Buffer-Insertion-Ring", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 67-68.

[Leung et al., 1990] W.H. Leung, T.J. Baumgartner, Y.H. Hwang, M.J. Morgan, and S.C. Tu,"A Software Architecture for Workstation Supporting Multimedia Conferencing in Packet Switching Network" , IEEE Journal on Selected Areas in Communications, vol.8, No.3, April, 1990, pp. 380-390.

[Linton et al., 1992] M.A. Linton, P.R. Calder, J.A. Interrante, S. Tang, and J.M. Vlissides,"InterViews Reference Manual Version 3.1 ", Stanford University, Dec., 1992.

[Little, 1992] T. Little, "Protocols for Bandwidth-Constrained Multimedia Traffic", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 47-48.

[Little and Ghafoor, 1990a] T. Little and A. Ghafoor,"Network Considerations for Distributed Multimedia Object Composition and Communications", IEEE Network Mag., Nov., 1990, pp. 32-49.

[Little and Ghafoor, 1990b] T. Little and A. Ghafoor,"Synchronization and storage Models for Multimedia Objects", IEEE JSAC, vol.8, No.3, April 1990, pp. 413-427.

[Little and Ghafoor, 1991] T. Little and A. Ghafoor,"Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks", IEEE Computer Mag. Oct., 1991, pp. 42-50.

[Loeb et al., 1992] S. Loeb, R. Hill, and T. Brinck, "Lessons from LyricTime, A Prototype Multimedia System", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 35-36.

[Loftin, 1993] R. Bowen Loftin, "Virtual Environment Technology for Aerospace Training", Virtual Reality System, Vol. 1. No. 2, Oct., 1993, pp. 36-37.

[Mitsuhashi et.al., 1987] K. Mitsuhashi, H.Shimizu, and S. Tsuruta, "Multimedia Communication Terminal for Network", IEEE/IEIEC Globecom'87, Tokyo, Japan, Dec., 1987, pp. 19.2.

[McAvinney, 1990] P. McAvinney, "Telltale gestures: 3-D applications need 3-D input",Byte, pp. 237-240, Jul., 1990.

[MIT, Release 5] MIT, "X Window System, Version 11", Release 5, 1989.

[Miyamoto et al., 1992] Y. Miyamoto, M. Ohta, and M. Yano,"A Multimedia System Based on MPEG Video Coding Algorithm", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 18-19.

[Mogal, 1992] J. Mogal, "The SGI Reality Engine", Panel on Virtual Reality, CyberArts International. Pasadena, CA, Oct.,1992.

[Morse, 1989] D. Morse, "The GNU Make Manual", Stanford University, Aug. 1989.

[Neapoulous et al., 1986] J-L. Neapoulous, P. Perron and A. R. Lecours,"The Biological Foundations of Gestures: Motor and Semiotic Aspects", Lawrence Erbaum Assoc. Pub. Hillsdale, New Jersey, 1986, pp. 11 -20.

[Nicolaou, 1990] C. Nicolaou, "An Architecture for Real-time Multimedia Communication Systems", IEEE Journal on Selected areas in Communicaitons, vol.8, No.3, April, 1990, pp. 391-400.

[Pesch, 1991] R.H. Pesch,"The GNU Binary Utilities", Oct., 1991.

[Petroni et al., 1991] M. Petroni, C. Collet, N. Fumai, K. Roger, F. Groleau, C.Yien, A Malowany, F. Carnevale, and R. Gottesman, "An Automatic Speech Recognition System for Bedside Data Entry in the Intensive Care Unit", Proc. Fourth Annual IEEE Symposium on Computer-Based Medical Systems, Baltimore, MD, May, 1991,

pp. 358-365.

[Pimentel and Teixeira, 1993a] Ken Pimentel and Kevin Teixeira,"Virtual Reality Through the New Looking Glass", Intel/Windcrest/ McGraw-Hill, Inc, 1993, pp. xiii-xvii.

[Pimentel and Teixeira, 1993b] Ken Pimentel and Kevin Teixeira, Virtual Reality Through the New Looking Glass, Intel/Windcrest/ McGraw-Hill, Inc, 1993, pp. 80-92.

[Pimentel and Teixeira, 1993c] Ken Pimentel and Kevin Teixeira, Virtual Reality Through the New Looking Glass, Intel/Windcrest/ McGraw-Hill, Inc, 1993, pp. 76.

[Rangan et al., 1992] P.V. Rangan, S. Ramannathan, H.M. Vin, and T. Kaeppner, "Media Synchronization in Distributed Multimedia File System", Computer Communication Review, Vol.22, No.3, July, 1992, pp. 88-89.

[Rosenberg, et al. 1991] J. Rosenberg, G. Cruz and T. Judd, "Presenting Multimedia Documents Over a Digital Network", Proceedings 2nd International Workshop onNetwork and Operating Support for Digital Audio and Video, Heidelberg, Germany, Nov. 1991.

[Recker, 1993] Rod Recker, "A System for Realistic Rendering of Virtual Architectural Environments", The Official Proceedings of Virtual Reality Systems '93", Conference and Exhibition, New York, Mar. 15-17, 1993, pp. 21-24.

[Rousseau, 1993] David Rousseau, " 3-D Display Controls for Sonar Operators", Virtual Reality System, Vol. 1. No. 2, Oct., 1993, pp. 28-32.

[Sachs, 1990] E. Sachs, "Coming soon to a CAD lab near you", Byte, Jul., 1990, pp. 238.

[Sakurai et al., 1990] Y. Sakurai, N. Ido, S. Gohara, and N. Endo,"Large Scale ATM Multi-stage switching network with shared buffer memory switches", Proc. ISS'90, Vol. 4, 1990, pp. 121-126.

[Sandrasegaran, 1993] K. Sandrasegaran, W. Cheung, R. Hossein, V. Zia, and A. Malowany, "A Computer-Based Learning Environment for Electronic Fault Diagnosis, Canadian Conference on Electrical and Computer Engineering, Sept. 14-17, 1993

[Satava, 1993] Richard M. Satava, "The Role of Virtual Reality in Medicine of the 21st Century.", Virtual Reality System, Vol. 1. No. 2, Oct, 1993, pp. 65.

[Schooler and Casner, 1992] E.M. Schooler, and S.L. Casner,"An Architecture for Multimedia Connection Management",Computer Communication Review, Vol.22, No.3, July, 1992, pp. 73-74.

[Shimoga, 1992] K. Shimoga,"Finger Force and Touch Feedback Issues Dextrous Telemanipulation", Proceedings of NASA-CIRSSE International Conference on Inteleligent Robotic Systems for Space Exploration, Troy, NY, Sept., 1992.

[Stallings, 1991] W. Stallings, "Data and Computer Communication Networks", Macmillan Publishing Company, New York, 1991, pp. 456-463.

[Steinmetz, 1990] R. Steinmetz, "Synchronization Properties in Multimedia System", IEEE JSAC, Vol.8, No.3, April,1990, pp. 401-412.

[Stone, 1991] R. Stone, "Advanced Human-System Interfaces for Telerobotics Using Virtual Reality Telepresence Technologies", Proceedings of the Fifth International Conference on Advanced Robotics ('911CAR), Pisa, Italy, June, 1991.

[Szuprowicz, 1992] Bohdan O. Szuprowicz, "Multimedia Techmology Combining Sound, Text, Computing, Graphics and Video, Computer Technology Research Corp., 1992, pp. 45-74.

[Tanabe et al., 1991] S. Tanabe, and K. Ohtsuki,"A Hyper-Distributed Switching System Architecture for B-ISDN", Proceedings ICC'91, 1991, pp. 44-45.

[Thomas and Stuart, 1993] John Thomas and Rory Stuart, "Speech Technology and Virtual Reality", Virtual Reality System, Vol. 1. No. 1, Mar., 1993, pp. 53-57.

[Vin et al., 1991] H.M. Vin, P.T. Zellweger, D.C. Swinehart, and P.V. Rangan, "Multimedia Conferencing in Etherphone Environment", IEEE Computer, vol24, No.10, oct., 1991, pp. 69-79.

[Vivid, 1993] The Vivid Group. Toronto, Canada, "Application Notes: The Mandala Virtual Reality Authoring System", Virtual Reality System, Vol. 1. No. 1, Mar., 1993, pp. 7.

[Vlissides, 1991] J. Vlissides and C. Brauer, " Ibuild User's Guide", Apr., 1991

[Wirtz and Maggioni, 1993] Brigitte Wirtz and Christoph Maggioni, "ImageGlove: A Novel Way to Control Virtual Environments", The Official Proceedings of Virtual Reality Systems '93", Conference and Exhibition, New York, Mar. 15-17, 1993, pp. 7-12.

[Weinstein and Forgie, 1983] C. Weinstein and J. Forgie, "Experience with Speech Communication in Packet Networks", IEEE JSAC, Vol. 1, No. 6, Dec., 1988, pp. 963-980.

Appendix A Listing of Upgraded ACME Code

The InterViews version 3.1 enhancements appear in the InterViews documentation. In InterViews Version 3.1, the included files action.h, canvas.h, button.h, glue.h have been upgraded. Hence the ACME client library has to be upgraded to permit compiling the ACME executable file, acme, in this new environment. The following changes were made.

In fileio_gui.c of client library,

change :

. *ActionCallbackdeclare(FILE_READER_PANEL)*

to :

. declareActionCallback(FILE_READER_PANEL)

change:

. *ActionCallbackimplement(FILE_READER_PANEL)*

to:

. implementActionCallback(FILE_READER_PANEL)

change:

. *FILE_READER_PANEL::FILE_READER_PANEL(FILE_READER* f, Kit* k, Style* s)*

to:

. FILE_READER_PANEL::FILE_READER_PANEL(FILE_READER* f, WidgetKit* k, Style* s)

Change:

. *Glyph* hspace = new HGlue(36.0)*

to:

. Glyph* hspace = new Glue(nil,36.0,nil, nil,nil)

Change:

```
. kit -> simple_push_button(
.   "rewind",
.    style,
.   new ActionCallback(FILE_READER_PANEL)(
.     this,
```

```
.          ActionMemberFunction(FILE_READER_PANEL)*)&FILE_READER_PANEL
:: rewind
.          )
.        )
to:
.      kit -> push_button(
.        "rewind",
.        new ActionCallback(FILE_READER_PANEL)(
.          this,
.          FILE_READER_PANEL::rewind
.          )
.        )
```

Change:

```
.      kit -> simple_push_button(
.        "loop",
.        style,
.        new ActionCallback(FILE_READER_PANEL)(
.          this,
.          (ActionMemberFunction(FILE_READER_PANEL)*)&FILE_READER_PANEL
:: toggle_loop
.          )
.        )
to:
.      kit -> push_button(
.        "loop",
.        new ActionCallback(FILE_READER_PANEL)(
.          this,
.          FILE_READER_PANEL::toggle_loop
.          )
.        ).
```

In acme_gui.c of client library,

Change:

```
.      ActionCallbackdeclare(SPEAKER_PANEL)
.      ActionCallbackimplement(SPEAKER_PANEL)
to:
.      declareActionCallback(SPEAKER_PANEL)
.      implementActionCallback(SPEAKER_PANEL)
```

Change:

. *SPEAKER_PANEL::SPEAKER_PANEL(ACME_RPC* a, int ld, int lt, int st, Style* s, Kit* k)*
to:
. SPEAKER_PANEL::SPEAKER_PANEL(ACME_RPC* a, int ld, int lt, int st, Style* s, WidgetKit* k)

change:
. *volume = new FieldEditor("10", s);*
to:
. volume = new FieldEditor("10", k, s);

change:
. *Glyph* hspace = new HGlue(12.0)*
to:
. Glyph* hspace = new Glue(nil,12.0,nil, nil,nil)

Change:
. *kit − > simple_push_button(*
. *"pause",*
. *style,*
. *new ActionCallback(SPEAKER_PANEL)(*
. *this,*
. *(ActionMemberFunction(SPEAKER_PANEL)*) &SPEAKER_PANEL :: pause*
. *)*
. *)*
to:
. kit − > push_button(
. "pause",
. new ActionCallback(SPEAKER_PANEL)(
. this,
. SPEAKER_PANEL::pause
.)
.)

Change:
. *kit − > simple_push_button(*
. *"softer",*
. *style,*
. *new ActionCallback(SPEAKER_PANEL)(*
. *this,*
. *(ActionMemberFunction (SPEAKER_PANEL) *) &SPEAKER_PANEL :: softer*
. *)*

```
.       )
to:
.     kit – > push_button(
.       "softer",
.       new ActionCallback(SPEAKER_PANEL)(
.         this,
.         SPEAKER_PANEL::softer
.         )
.       )


   change:
.     kit – > simple_push_button(
.       "louder",
.       style,
.       new ActionCallback(SPEAKER_PANEL)(
.         this,
.         (ActionMemberFunction (SPEAKER_PANEL) *) &SPEAKER_PANEL :: louder
)
.       )
to"
.     kit – > push_button(
.       "louder",
.       new ActionCallback(SPEAKER_PANEL)(
.         this,
.         SPEAKER_PANEL::louder
.         )
.       )
```

In the acme_gui.h included head file, the SPEAKER_PANEL provides an interface for pausing and controlling the volume of an ACME speaker.

```
   Change:
.     class SPEAKER_PANEL : public LRBox
to:
.     class SPEAKER_PANEL : public Box


   change:
.     Xit* kit
to:
.     WidgetKit* kit
```

In the fileio_gui.h included head file, A FILE_READER_PANEL provides but-

tons to rewind to the beginning, and to toggle the rewind-at-EOF mode.

change:
. *class FILE_READER_PANEL : public LRBox*
to:
. class FILE_READER_PANEL : public Box

change
. *Kit* kit*
to
. WidgetKit* kit

change
. *FILE_READER_PANEL(FILE_READER* f, Kit* k, Style* s)*
to
. FILE_READER_PANEL(FILE_READER* f, WidgetKit* k, Style* s)

Appendix B Example of Member Function Implementations in the User Interface Test Program

```
#include < InterViews/button.h >
#include "vl.h"
#include < IV-2_6/_enter.h >
int pidsv = 0;
int pidcl = 0;
int pidex = 0;
extern int pid;
extern int pidla;
extern int pidsa;


    /* Class vl is derived from class vl_core */
vl::vl(const char* name) : vl_core(name) {}


    /* svided, cleared, and exited are vl member functions */
void vl::svided() { /* member function implementation for Send Video button*/

    if (pidsv = fork() == 0) { /* create a new process */


    putenv("ACME=haley"); /* setting the ACME environment to the remote work-
station */

    /* play a video sequence from the local workstation to a remote workstation */

    system("src/apps/studio/studio src/apps/studio/script.playback");

    svidBS – > SetValue(0); /* deselect the Send Video button after finishing the
video playback */

    }
}


    /* member function implementation for Clear button */
void vl::cleared() {

    if (pidcl = fork() == 0) { /*create a new process */
```

```
    system("clear"); /* system call to clear video display window */

    clearBS – > SetValue(0); /* deselect the Clear button after finishing the window
clearing */

    }
}


    /* member function implementation for Exit button */
void vl::exited() {

    char s[256];

    printf ("%d %d %d
n", pid, pidcl, pidex);

    /* send the terminate signal to the specified process IDs, pid1, pid2, pid3, pid4
and pid5. */

    sprintf(s, "kill %d %d %d %d %d ", pidcl, pid, pidla, pidsv, pidsa); /* exit the
test program */

    printf("system: %s", s);

    if (pidex = fork() == 0) { /* create a new process */

    system(s); /* system call to exit the test program user interface */

    exitBS – > SetValue(0); /* deselect the Exit button after exiting the test program
*/

    }
}
```

The related "script.playback file containing:

```
video_out /home/rain/yifan/acme/acme_data/miss_america 0 100000000 400 200
352 288
video_out /home/rain/yifan/acme/acme_data/football 0 166000000 416 524 320
240
audio_out /usr/demo/SOUND/sounds/spacemusic.au 0 3000 100
audio_out soundtracka 0 16600 200
```