2009

Surgnet: An Integrated Surgical Data Transmission System over Collaborative Networks

Sriram Natarajan
University of Massachusetts Amherst

SURGNET: AN INTEGRATED SURGICAL DATA TRANSMISSION SYSTEM OVER COLLABORATIVE NETWORKS

A Thesis Presented

by

SRIRAM NATARAJAN

Submitted to the Graduate School of the University of Massachusetts Amherst

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

February 2009

Electrical and Computer Engineering

SURGNET: AN INTEGRATED SURGICAL DATA TRANSMISSION SYSTEM OVER COLLABORATIVE NETWORKS

A Thesis Presented

by

SRIRAM NATARAJAN

Approved as to style and content by:

____
Aura Ganz, Chair

____
C. Mani Krishna, Member

____
Ramgopal R. Mettu, Member

____
Christopher V.Hollot, Department Chair
Electrical and Computer Engineering

DEDICATION

To,

My Parents and My Brother

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Aura Ganz for helping me through the rigors of starting a research topic from the scratch and also for her guidance, suggestions and support throughout my graduate studies. It would have been impossible to bring my ideas and project to realization without her help. I thank her for giving me an opportunity of being a part of her group and giving me the time and space for the realization of the project. My graduate education has been a learning experience in all aspects of my life.

I would like to thank Prof. Mani Krishna and Prof. Ramgopal Mettu for being a part of my thesis committee and for guiding me through my thesis.

I also would like to thank all my friends, both from MNL and outside (Gautam Das, Shiva, Russell, Kartik, Vivek, Samant, Shashi, Pritish, Anand, Sudarshan) for their help and support in different ways. They were instrumental in providing a cooperative and mutually encouraging atmosphere both in the lab and outside. Their constant guidance, constructive criticism and above all their belief in me during the difficult initial stages of my graduate school research was a great source of motivation.

Finally, I would like to thank my family, my mother, father and brother, for their constant encouragement and belief in me. They have always been there for me and offered unconditional love and support to me. I owe them everything I have achieved.

ABSTRACT

SURGNET: AN INTEGRATED SURGICAL DATA TRANSMISSION SYSTEM OVER COLLABORATIVE NETWORKS

FEBRUARY 2009

SRIRAM NATARAJAN

M.S., UNIVERSITY OF MASSACHUSETTS, AMHERST

Directed by: Professor Aura Ganz

Telesurgery relies on fast and reliable data transmission between the surgeon and tele-operator side over lossy and delay constrained networks. Medical data involves audio, video, ECG and Force Feedback data. When these media streams are transmitted through best effort networks, the temporal information gets affected due to network constraints. Major network degradation is due to the Force Feedback device with rendering rate of 1 KHz, hence data is generated every millisecond. In our proposal we concentrate on improving the synchronization of force feedback device on varying networking conditions.

Force feedback data is generated by operating a source (surgical) device which controls the movement of remote device. It has a great potential in improving telemedicine facilities, when included with the support of different multimedia services. The channel imposes delay and packet loss constraints for such devices which require unique solutions, unlike audio or video media, due to its high rendering rate.

Current research supports Force Feedback in fiber optic communication, packet switched networks. However, such schemes are not feasible in supporting surgical telepresence system. While efforts are made to support force feedback media in wireless medium, few works have addressed delay synchronization and loss of data. There exists no previous work which has attempted to provide an efficient integrated solution where video and force feedback information have been supported by the same network. This thesis focuses in providing an integrated architecture that supports the force feedback data over a collaborative network and improves the data synchronization and packet loss prediction in the remote side over a varying network link. The goal will be to evaluate the support of such data types.

We have implemented a Linear Packet Predictor Algorithm which predicts the missing packet value. Data generated from the source device are sent as UDP packets. UDP transmission is unreliable and hence we use an RTP over UDP to make it reliable. Each packet will have the current position of the device and force applied. We use a Microsoft Sidewinder Force Feedback joystick. The handle of the joystick is located at the center of the base. So we record the position of the device on both positive and negative axis moving in a two dimensional space. This device provides rotational movement and hence drastic change in position occurs within milliseconds. Once the packet arrives at the receiver side, the control unit checks for the sequence number of the packet. If continuity is missing then, the control unit passes the packet to the predictor algorithm which predicts the packet else it directly updates the packet to the Virtual Time Rendering Algorithm

Another major issue is the delay jitter. On the source (server) side the intra time difference between two packets will be 1msec. But due to varying delay in the network the data packets arrive at the receiver with fluctuating intra time difference. In order to counter the delay jitter effect, we implement the Virtual Time Rendering algorithm which reads the time stamp value at which the packet was generated at the source and modifies the update time at the receiver side. In our work we do not control another device on the remote side, rather an applet which was developed using a Virtual Reality Markup Language in Matlab.

Another challenge which is imposed when other multimedia is introduced with force feedback is the intra media synchronization. Real time video is captured from the applet side and given as feedback to the server side to improve the interactivity of the application. At every instant in time, different multimedia data produce data to be updated at the remote end. Since all the information are inter dependent with other media in time, efficient intra media synchronization is required.

This thesis also focuses in providing an architecture which not only supports force feedback data but have a multiplexed model which allows an efficient transmission of all surgical information in real time. Each data occupies significant part of bandwidth in the network and the effect of multiplexing might affect the synchronization scheme of the force feedback device. Our architecture supports the efficient transmission of all types of multimedia information and also maintain the synchronization of the scheme. This method is unique with its methodical approach to support different multimedia information.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Current day Internet supports to carry various forms of multimedia information. A multimedia system is characterized by the integrated computer controlled generation, presentation and communication of independent discrete and continuous media data. The presentation of various data such as the audio, video, medical data and force feedback data requires synchronization between these media which is the key issues of an integrated transmission. Many applications stream live data from remote areas like news updates, weather forecast, and military data updates to the remote station. Multimedia systems have to coordinate the relationship among all media. These relationships include the temporal and spatial relationships. Temporal relationships are the presentation schedule of media and spatial relationship are the location arrangement of the media.

The temporal relationship and the multimedia synchronization mechanism ensure the temporal ordering of events in the multimedia systems. Multimedia synchronization can be distinguished into three types: intra-stream synchronization, inter-stream synchronization and inter-media synchronization.

Intra-stream synchronization are the play-out synchronization which ensures that the receiver plays out the media at a fixed time after it was generated at the source and experienced variable end to end delay. It ensures that the constant rate source at the sender becomes a constant rate source at the receiver despite the delay jitter due to the network. For example a live feed video data with a rate of 30 frames per second, each of the frames must be displayed for 20ms. If the arrival rate of the video data is modified due to the time varying network with delay jitter, then this affects the presentation of the video frames at the receiver which causes a jittery video picture at the receiver. Intra stream synchronization is the base part of various video coding such as the H.261 and MPEG coding systems.

Inter stream synchronization is when all receivers play the same segment of the medium at the same time. This is required in collaborative networks where there are several receivers. Each receiver receives update from the single source and receives the stream at different instance of time. But the synchronization scheme should ensure that the streams are played in sequence as it was generated in the source. In a collaborative session the same media information may be reacted upon by several participants. In our study we do not consider multiple players in the receiver side. We maintain a single user/receiver and single source which collaborates information. So we do not deal with the intra stream synchronization.

Intra media synchronization is concerned with maintaining the requirements of the temporal relationships between two or more media. Lip synchronization is an example of intra media synchronization where the display of video must be synchronized with the audio. This is required in various applications such as TV broadcasting, Films, Audio channels, Web casting and video hosting sites. For example in Youtube we see that many videos lack the synchronization between the video and audio due to Lip Sync error. This work is done in post production in most applications by adjusting the playout time of the audio with the video content.

In distributed environment some of the time factors cause asynchrony among the media. (Akyildiz and Yen, 1996):

- *Different Skew:* There is a time difference between media objects at the destination after the arrival of these packets (audio and video) through time varying delay oriented networks. Hence they are out of synchronization at the receiver side
- *Different Jitter:* On a communication channel there is maximum different end to end delay experienced by any two consecutive objects. Due to the delay jitter the media objects may arrive before or after its playback time.
- *Different Initial Collection Time :* When there are several media senders in the distributed communication, these senders must collect media objects and transmit them synchronously, else the temporal relationships among media objects might be affected

In a distributed multimedia environment, at the destination the presentation components need to have the synchronization specifications information at the moment an object is to be displayed. There exists three approaches for the delivery (Blakowski and Steinmetz, 1996)

- *Delivering the complete synchronization information before the start of the presentation:* This approach is often used in presentation and retrieval based systems with stored data objects that are arranged to provide new combined multimedia objects.

- *Using an additional synchronization channel:* This approach is used in live synchronization when the synchronization information is available only at the time the multimedia data is being captured. Because of additional channel used it is not very feasible for the case of multiple sources.
- *Multiplexed Data Streams:* In this approach the multimedia data and the related synchronization information is multiplexed on one communication channel and delivered together. This is suitable for different media at the same source location and not in distributed media environments

1.1 GOAL OF PROPOSED THESIS WORK

The primary goals of SURGNET are:

- ***Intra Stream Synchronization***: Intra stream synchronization involves the playout time modification of the data stream which arrives at the receiver. Unique challenge is faced when we work on the Force feedback device. As mentioned these devices render at 1 KHz and hence generate data every 1 millisecond. Data generated at the source will carry the time stamp information at which time the data was generated. Due to *Different Skew* and *Different Jitter* issues as discussed in Section 1 the data arrive at the receiver with modified time instants. Random delay values are possible between media streams and hence a deterministic solution cannot be provided to modify the synchronization criteria.

Major issues that modify the packets (here data streams and packets are used interchangeably) are the packet loss and delay jitter. In our study we use an emulated network which introduces Derivative Random Drop (DRD) data loss and variable delay modifications (Heavy Tail Distribution).

In our experiment we use a Microsoft Sidewinder Force feedback Joystick. The device interacts with the user via sense of touch by applying force, vibrations and motions to the user. The mechanical stimulation is used in creating virtual objects that exist in computer simulations to control the virtual objects which would enhance the remote control of machines and surgical devices located at remote stations through networks. The emerging technology promises to support telesurgery. This information is transmitted through an emulated network environment using NISTNet Emulator. The emulator introduces delay constraints and packet losses by which the data packets are modified from its actual generation times. The packet reaches the destination at particular receiver time with modified data.

The client runs a Virtual Reality Applet developed in Matlab which runs on a web browser. The applet consists of crane model which contains a moving device controlled by our source joystick through simulated network. We have developed compensation techniques which modify the data packet timings and predict lost packet data and then update the applet with timings as generated at the source.

Live real time video of the applet is captured at the client and sent to the server to improve the interactivity of the application. On the server side, Image analysis such as brightness modification, gray scale image generation, image rotation and zooming are performed so as to perform detailed analysis on the captured data. The movement of the force feedback device is inter dependent with the image analysis in this collaborative application.

Synchronization is required to improve the quality of transmitting the force feedback data through network. The scheme involves an integrated solution to counter packet loss and delay jitter. We implement two efficient algorithms to counter the network impacts

- *Linear Packet Predictor Algorithm:* Linear Prediction is a mathematical operation where future values of discrete time signals are estimated as a linear function of previous values. The predicted signal is calculated based on the prediction coefficient which determines the number of previous values to be included in order to calculate the missing value or the future value. The error generated by the system is the difference in the actual missing value to the predicted value. The most common estimation of parameters is the root mean square criterion also referred as the autocorrelation criterion. We minimize the expected value of the squared error. We consider a single dimension in determining the expected value.

- *Virtual Time Rendering Algorithm:* Media Packet generated at the source carries a timestamp value with every packet. The time stamp indicates the generation time of the packet. The temporal relationship such as the generation time should be maintained at the receiver side. The data packets arrive at the receiver side which is denoted as the Arrival time. This data will have a modified time difference between continuous packets. We compare the timestamps of consecutive packets with the timestamp difference of two consecutive packets at the source. This by generation is always 1 millisecond. The arrived time difference between two packets might be less than 1 millisecond or more than 1 millisecond. If the difference is less then we need to virtually expand the time rendering rate between packets so as to bring it to 1 millisecond. If the timestamp difference is higher than 1 millisecond then we need to virtually contract the timestamp difference so as to synchronize as it was generated in the source. This is achieved efficiently by the Virtual Time rendering Algorithm which adjusts the rendering rate before updating the applet which is controlled by the source joystick.

By efficiently integrating the two algorithms we achieve the intra stream synchronization among force feedback device and the applet running on different locations. The video, image data received in the reverse from the client increases the end to end delay but the algorithm we developed is independent of the end to end delay and maintains the accuracy of the application.

The rest of the proposal is organized as follows: Chapter 2 provides a background about force feedback devices and its usage in distributed applications. We will also discuss about the various synchronization schemes in existence and discuss about how efficiently SURGNET achieves in synchronizing force feedback data. Chapter 3 discusses about the SURGNET Architecture. Chapter 4 explores the architecture in detail with Server, Network and Client Architectures. Our testbed and comparative results are discussed in Chapter 5. Chapter 6 briefs about the future work and Chapter 7 concludes this proposal

CHAPTER 2

BACKGROUND

2.1 FORCE FEEDBACK DEVICE

Force feedback or Haptic refers to technology which interfaces the user via the sense of touch by applying forces. This emerging technology promises to have wide reaching applications. Haptic technology has made it possible to investigate in detail how the human sense of touch works, by allowing the creation of carefully-controlled haptic virtual objects. [11], [12], [13]. These objects are used to systematically probe human haptic capabilities.

Haptic interfaces are used for medical simulation may prove especially useful for training of minimally invasive procedures (laparoscopy/interventional radiology) [14] and remote surgery using teleoperators [15]. Such devices will be used in future in Telepresence applications whereby an expert surgeon may work from a central workstation, performing operations in various locations, with machine setup and patient preparation performed by local nursing staff. A particular advantage of this type of work is that the surgeon can perform many more operations of a similar type, and with less fatigue. It is well documented that a surgeon who performs more procedures of a given kind will have statistically better outcomes for his patients.

A 'Virtual Haptic Back' (VHB) was successfully integrated for students at the Ohio University College of Osteopathic Medicine [16]. Research indicates that VHB is a significant teaching aid in palpatory diagnosis (detection of medical problems via touch). The VHB simulates the contour and compliance (reciprocal of stiffness) properties of human backs, which are palpated with two haptic interfaces (SensAble Technologies, PHANToM 3.0).

In our experiment we use the Microsoft Sidewinder Joystick Force feedback which provides the required positional and force values when we operate the device. We found that motor control was accomplished through the use of two controllers, one for each motor. The heart of each controller is a driver chip, which outputs a voltage to a motor based on a pulse-width-modulated signal from the control board processor. The position of the joystick along each axis is determined using general purpose differential comparators, which compare the voltage of two potentiometers against the reference voltage. The internal processor was found to run at 25 MHz.

Figure 2.1 Microsoft Sidewinder Force Feedback Joystick

The active force feedback joystick can serve as a robotic manipulator or a haptic simulation input device. Since the device is actuated by two dc motors, which control the stick in perpendicular directions, the joystick can also be used for servomotor position control experiments. Our developed architecture features an ActiveX controller which interfaces directly with Matlab and Simulink. ActiveX is a programming methodology developed by Microsoft to allow software components to interact with each other and the operating system on the same machine or over the Internet [17]. These components are created with any modern windows compiler and do not require the installation of additional software.

To make the device communicate with the Simulink model we use the ActiveX to interact with our architecture as a source or a sink, without having to know the details of how these components were coded. In the Virtual Reality Tool kit supported by the Simulink we can develop our external components to either act as a source or a sink. When the device acts as a Source, information about the state of the joystick is available as an input to a Simulink block diagram. When acting as a sink, the controller accepts two inputs that represent the force to be applied to each axis. We also use a rendering engine based on DirectX9 which, when integrated with Matlab controls the movement of the joystick with the updated position and force feedback from the remote side. Internal Video capture is activated to record the movements of the applet and sent to the server side. The server then performs the Image analysis for improving the intractability of the application which will decide the further movement of the device.

2.2 INTRA STREAM SYNCHRONIZATION

A major feature which distinguishes multimedia communications from other traditional ones is simultaneous handling of plural media streams which are temporally related to each other. In multimedia communications, we therefore have to take account of the temporal structure within a single media stream and that among the plural media streams. However, media capturing process and network delay jitters disturb the temporal structure. Therefore, we need media synchronization control, which preserves the structure. A number of media synchronization algorithms have been proposed to meet diverse requirements. Media synchronization control can be classified into four different groups such as the Basic Control, Preventive Control, Reactive Control and Common Control. The efficiency of the algorithm can be compared with other algorithms based on clocks, advance information on network delay bounds, locations and synchronization control techniques, which determine the design of each algorithm. The four kinds of communication patterns among one or more sources and one or more destinations. They are 1 : 1**,** *n:* 1, 1 *:n* and *n:m.* In our application we work with a single client and single server architecture.

2.3 REACTIVE CONTROL

2.3.1 SOURCE CONTROL

Adjustment of transmission timing:

The transmission timing of MUS is adjusted according to the amount of the skew among media streams [19]. When the destination detects a skew, it sends feedback information to the source to change the transmission timing. The source performs this action by changing the transmission period. This is not suitable for our real time requirement.

Decrease of the number of media streams transmitted:

The source decreases the number of media streams transmitted when it is difficult for the destination to recover from asynchrony [20], [21]. For instance, in lip synch, the source can stop the transmission of the video stream temporarily; when the destination recovers from asynchrony, the source can restart the transmission.

2.3.2 DESTINATION CONTROL

Reactive skipping (discarding) and reactive pausing (repeating):

When the output timing of the current MU is late, the destination skips the succeeding MUS if the MUS arrives earlier [21-39]. We can also discard late MUS [40], [41]; however, discarding late MUS may lead to poor average MU rate, which is defined as the average number of MUS output per second at the destination. When the buffer starvation occurs, the destination pauses output of MUS. This means that for video the destination continues outputting the previous MU until the next MU becomes available. It is also possible to output another data at this time. These techniques are the most popular since it is easy to implement them.

Shortening and Extension of output duration: In order to recover from asynchrony gradually without large degradation of the output quality (that is, so as not to be noticed by users), the destination shortens or extends the output duration of each MU until the recovery from asynchrony [24]-[39]. Shortening of the output duration of MUS includes fast-forwarding (without skipping) of MUS, and extension leads to pausing output of MUS. The control can also be used to adjust the output timing of a media stream to that of another stream.

For voice, we can make use of silence period of voice to shorten or extend the output duration of MUS". Extension of the output duration is similar to preventive pausing since the latter can be realized by extending the output duration. However, the former is reactive, while the latter preventive

2.4 VIRTUAL TIME RENDERING

Virtual time-contraction and time-expansion: In addition to the actual time, we introduce a virtual time which expands or contracts according to the amount of delay jitters of MUS received at the destination, and media are rendered along the virtual-time axis. In [24] the virtual time-contraction and time-expansion are realized in a form of modification of the *target output time,* which is the time when the destination should output the data packet.

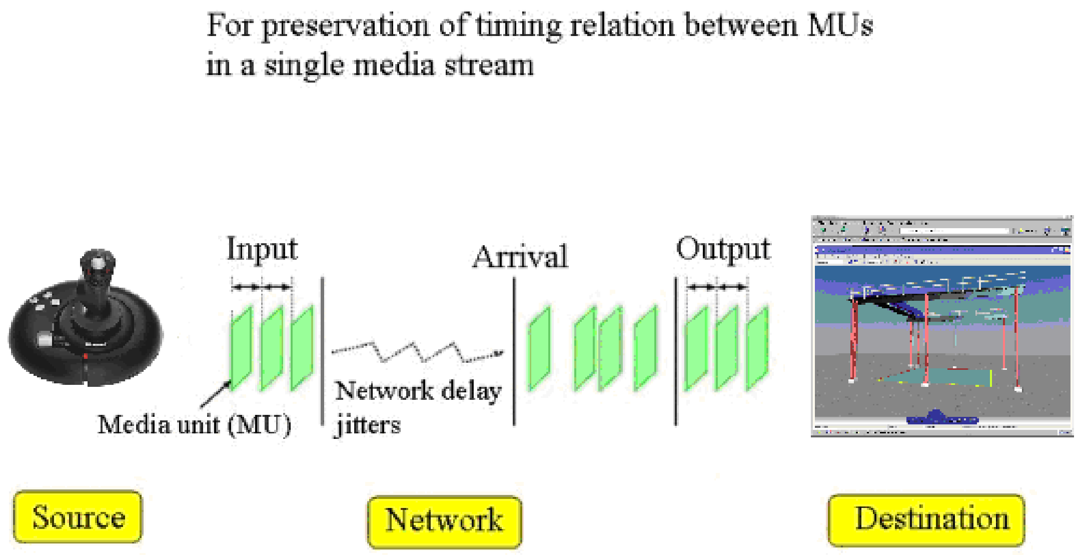


Figure 2.2: Intra Stream Synchronization

In [38], the *slide control* scheme, which changes the amount of the modification of the target output time according to the extent of asynchrony, is proposed and applied to PHS (Personal Handy Phone System) [38]. In [21], the Logical Time System (LTS) corresponds to the virtual time. Only virtual time-expansion is performed by stopping the LTS temporarily. Virtual time-expansion is also exploited in [42] and [43]. The set-back and advance operations of the PlayOut Clock in [44] correspond to virtual time-expansion and time-contraction, respectively.

This technique differs from shortening and extension of output duration in that the former indirectly changes the output timing by modifying the the virtual-time (equivalently, resetting the origin of the time axis), while the latter directly does (that is, the origin of the time axis is kept the same).

CHAPTER 3

ARCHITECTURE

3.1 SYSTEM ARCHITECTURE

The model of telesurgery application comprises of client server architecture as in Figure 3.1. The system comprises of a force feedback device (Microsoft Sidewinder Joystick) at the server, for the surgeon to perform the task under emulated virtual environment. The emulated commands and control information are fed forward in real time through the network to the remote process. A Virtual Reality enabled Browser applet runs on the Client Side. Updated position and force information from the joystick are used to control the movement in the applet.

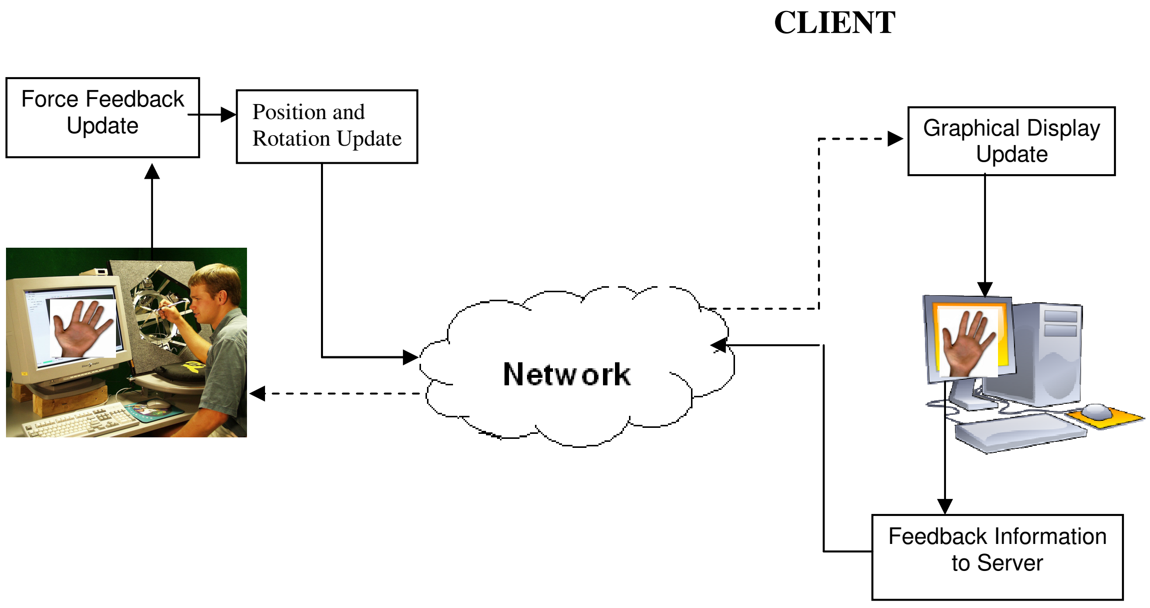


Figure 3.1 System Architecture

Regular updates are sent by capturing live video feedback using an internal cam recorder and sent to the server. Networks are prone to be lossy as well as delay constrained. Today's network carries best effort traffic without providing Quality of Service (QoS) to applications. Due to sharing of bandwidth most time dependent applications are error prone. Real time applications are mostly affected because of the time constrained nature. Force feedback devices generate data with rendering rate as high as 1 KHz. So data is generated every 1msec unlike other multimedia such as audio and video whose rendering rate is 30hz.

Input command from the force feedback device at the source is generated in the form of force media, which is segmented into sequenced data packets. The force data packets contains information about the updated X axis and Z axis position and force recalculation of the device. This information corresponds to the position of the operator hand movement when moving the device and will be sent to update the applet which consists of corresponding joystick position hanging in a crane.

Each data packet is generated every 1msec and sent to the emulated delay varying, lossy network. Due to delay variations simulated, each data packet is subjected to fluctuations upon arrival on the receiver side. The fluctuated arrival of the data packet causes change in the update rate of the applet. So the applet gets updated in a random manner which causes instability to the system.

To solve the delay variation and data loss from the network we need data synchronization algorithms. The major problems in the network are the data loss and delay jitter issues. Packet loss is a major issue due to multiple hops in the network. When networks become more complex, loss of packets becomes a major issue. To solve this problem we implement the existing Linear Packet Predictor Algorithm. Another major concern is the delay jitter. When the data comes from a delayed network and if we update the applet with the time the packet arrives it causes instability. So the time difference between the packets has to be maintained as how it was generated at the source.

In order to change the update rate at the receiver we implement a Virtual Time Rendering (VTR) Algorithm which either contracts or expands the time at which each packet has to be updated at the applet. Usually when data passes through wide area networks, there are fewer chances that time contraction would be required. Virtual Time Rendering Algorithm requires the time stamp information when the data was generated. So when we generate the data at the source, we generate the time stamp value for each packet. So when the data arrives at the receiver side, the VTR reads the time stamp values and modifies the rendering time of the packet at the applet. More details of the working of the algorithm will be discussed in next sections.

Video is captured and streamed constantly from the client side to the server. Standard compression techniques is used here inorder to reduce bandwidth. The media storage size is dependent on the streaming bandwidth and the length of the media. In our experiment we have recorded the video for testing purposes to improve the collaboration of the application and hence the capture rate is highly random. Imaging techniques will be discussed in detail in the client architecture section.

For a one minute recorded video which is sent to the server, the video encoded at 300 kbits/s (encoded in a 320 * 240 pixels window size).

Hence the size would be : (60s * 300 kbits/s) / (8*1024) = 2.19MB of storage

CHAPTER 4

DETAILED ARCHITECTURE

4.1 SERVER ARCHITECTURE

Data generated from the server side is transferred to the Client side through an emulated network. Server takes input from the Joystick and converts the input into a Matlab readable input file which acts as the source of data. Force data are updated in real time and has to be sent to the client side with time constraint. Hence we use UDP as the transport protocol which receives packetized data from the source and randomly without any guarantee sends it to the client side. Here in this section we discuss how the data is obtained from the joystick and how it is processed by the Matlab and packetized and sent via the emulated network. In the next section we discuss about the network architecture and followed by the Client Architecture.

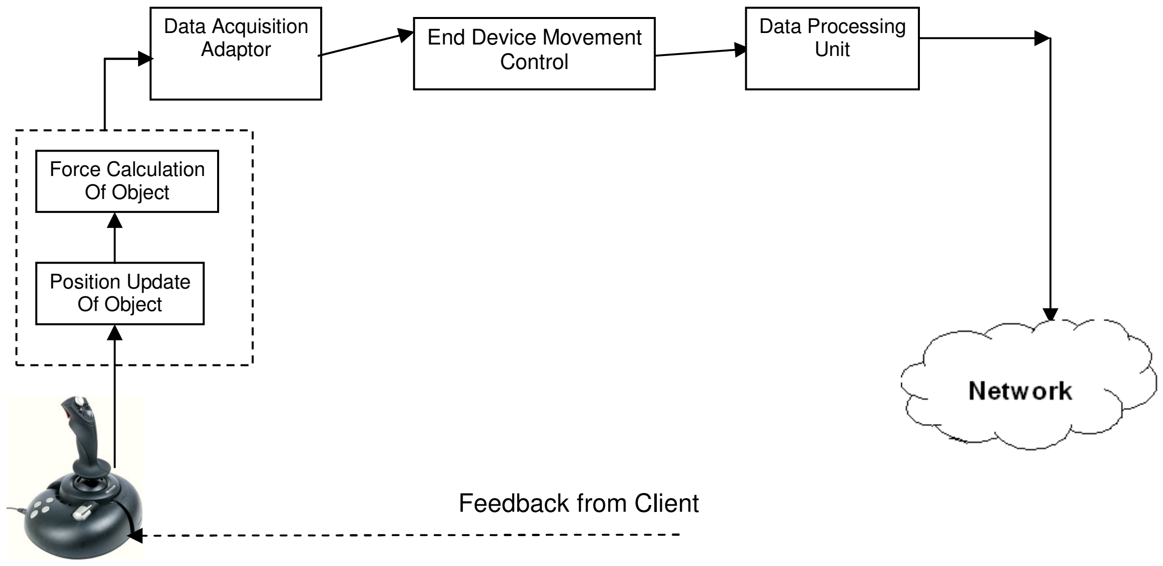

Data Acquisition Adaptor
End Device Movement Control
Data Processing Unit
Force Calculation Of Object
Position Update Of Object
Network
Feedback from Client

Figure 4.1 Server Architecture

4.2 FUNCTIONAL DESCRIPTION

4.2.1 SIDEWINDER JOYSTICK

Sidewinder joystick produces force feedback whenever the handle is gripped and proper forces are applied to the buttons. The device consists of eight programmable buttons which are set to produce different forces. The Joystick rotates in addition to conventional x-axis (side-to-side) and z-axis (back-and-forth) joystick movement. When the device rotates it produces values on both X and Z axis.

Force feedback data is generated at the server side and processed, to be sent to the received side. Sidewinder Joystick generates data every 1msec. Usually Sidewinder joystick is used for gaming purposes and hence it has some pre processed data which generates force data. But for our experiment purpose we use just one button which generates the force. The base of the device is fixed in the center and hence the device can move on either left and right on the X-Axis and up and down on the Z-Axis. We capture this information from the device manager which allows us to program the device in order to send the information to the data acquisition adapter in Matlab.

4.2.2 DATA ACQUISITION ADAPTER

Force positional data are received by the data acquisition adapter. It takes two inputs from the joystick. The main button is programmed to provide a feedback force which will move the joystick to the desired position. The axes takes inputs from both X and Z axis.

The maximum range for both the axes is set to -100 and 100 so that we can record the movements. Any movement which exceeds this value will be rounded to 100 in the positive range and -100 in the negative range. To our convenience we have set a constant value for the Y Axis.

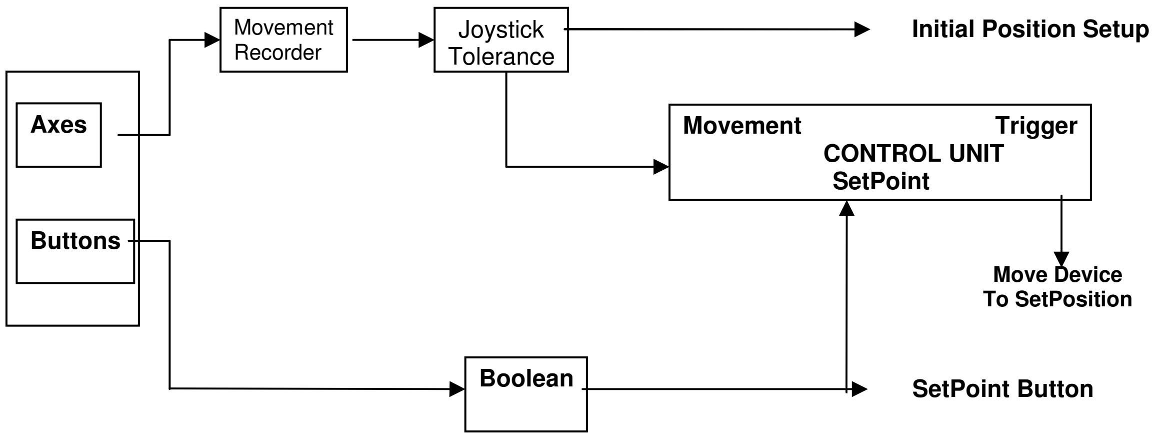


Figure 4.2 Data Acquisition Adapter

The applet is controlled by initially setting the position where the device has to be moved and then press the programmed button to move the actual position. In figure 2 we see that the data from the joystick axes (X and Z) as well as the button are obtained and sent to the processing unit.

The movement recorder controls the axes movement, records the real time position in both the axes and sends it to the joystick tolerance sends it to the joystick tolerance unit. The joystick tolerance sets the maximum, threshold value to which the joystick can move in either the X axis or the Z axis. Any value which exceeds 100 on the positive axis or -100 in the negative axis will be rounded to 100 and -100 respectively. This might create errors, but we are not concerned in recording the error of hardware here.

Button produces the force to be applied. The initial position is recorded and only when the button is pressed, the actual movement of the joystick in the applet takes place. Here the button works in as a digital format. It takes just two inputs, with a recorded force value. When the button is not pressed, it produces a 0 and when the button is pressed, it takes in a 1 which takes the force value and records that the joystick has to be moved.
The control unit takes in both the axis input and the button value. Initial position movement will be recorded and updated until the button is pressed. Once the control unit receives the button value, it actually starts moving the joystick to set it to the desired position.

4.2.3 DATA PROCESSING UNIT

The data processing unit packetizes the data into force data packets. Each data packet contains the current position and force value. Each data packet is generated every millisecond. A time stamp value is generated when every data packet is created and added to the header of the packet. Due to the real time nature of the application we cannot use TCP/IP as the transport protocol, because of its delayed, inefficiency to carry real time data. TCP/IP performs flow and error control and ensures no data is lost. It waits until all packets are arrived at the receiver. If any packet is lost, it protocol requests for retransmission of the packet. So by all means to maintain the efficiency of the application TCP/IP is not suitable.

UDP generates datagram carrying the same information. UDP is considered to be unreliable or does not maintain the ordering of packets as maintained by the TCP stack. Datagrams may arrive out of order, appear duplicated, or go missing without notice. Avoiding the overhead of checking whether every packet actually arrived makes UDP faster and more efficient, at least for applications that do not need guaranteed delivery. However Time-sensitive applications such as ours use UDP because dropped packets are preferable to delayed packets.

Lacking reliability, UDP applications must generally be willing to accept some loss, errors or duplication. Lacking any congestion avoidance and control mechanisms, network-based mechanisms are required to minimize potential congestion collapse effects of uncontrolled, high rate UDP traffic loads. In other words, since UDP senders cannot detect congestion, network-based elements such as routers using packet queuing and dropping techniques will often be the only tool available to slow down excessive UDP traffic.

To apply synchronization criteria, we need to have some reliability information about the data packets. Sequence Numbers and Time Stamp information are important to provide reliability to the application. We run the packet predictor algorithm which requires the sequence number information and Virtual Time Rendering which requires the Time Stamp Information. UDP packets do not carry this information. In order to provide such reliable information we run Real-time Transport Protocol (RTP).

RTP defines the standardized packet format for delivering audio and video over the internet. Since our force feedback information also falls in the real time criteria, we use the RTP implementation for the force packets which are generated in real time and updated at the receiver end

RTP does not have a standard port to communicate with the communicating party. The only standard that it obeys is that UDP communications are done via an even port and the next higher odd port is used for RTP Control Protocol (RTCP) communications.

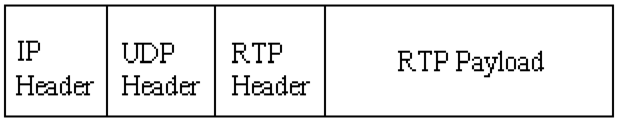


Figure 4.3 RTP Packet Format

The real-time media that is being transferred forms the RTP Payload. RTP header contains information related to the payload e.g. the source, size, encoding. RTP packet can't be transferred as it is over the network. For transferring we use a transfer protocol called User Datagram Protocol (UDP). RTP runs over UDP to make sure the synchronization schemes reads the data, extracts the packet information and performs the synchronization.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.4 Packet Structure

4.3 NETWORK ARCHITECTURE

4.3.1 NISTNET EMULATOR

The NISTNet network emulator is a general-purpose tool for emulating performance dynamics in IP networks. The tool is designed to allow controlled, reproducible experiments with network performance sensitive/adaptive applications and control protocols in a simple laboratory setting. By operating at the IP level, NISTNet can emulate the critical end-to-end performance characteristics imposed by various network situations.

NISTNet is implemented as a kernel module extension to the Linux operating system, running Fedora Core 2.6.11 kernel module and an X Window System-based user interface application. In use, the tool allows an inexpensive PC-based router to emulate numerous complex performance scenarios, including: tunable packet delay distributions, congestion and background loss, bandwidth limitation, and packet reordering / duplication. The X interface allows the user to select and monitor specific traffic streams passing through the router and to apply selected performance "effects" to the IP packets of the stream. In addition to the interactive interface, NISTNet can be driven by traces produced from measurements of actual network conditions. NISTNet also provides support for user defined packet handlers to be added to the system.

In our architecture we have emulated the network which emulates various networking conditions. NISTNet Emulator acts in-between the server and the client which receives the data packets generated by the server and introduce loss and delay to each individual packets. For our experiment we have created a Derivative Random Drop (DRD) to drop packets in a random fashion. Variable delay with heavy tail distribution is introduced which introduces random delay between each packet.

The emulator which implements this functionality consists of an instrumented version of a live network implementation, the Linux (2.0.xx – 2.4.xx) kernel. NISTNet consists of two main parts:

(1) a loadable ***kernel module***, which hooks into the normal Linux networking and real-time clock code, implements the run-time emulator proper, and exports a set of control APIs; and

(2) a set of ***user interfaces*** which use the APIs to configure and control the operation of the kernel emulator

This organization of the emulator provides several advantages. Since all of the kernel functionality is incorporated into a loadable module, the emulator may be taken up or down, patched and reloaded during runtime, without interrupting any active connections, including those flows currently being affected by the emulator. The separation into a module also serves to insulate the NISTNet code to a large degree from changes to the base kernel.

The separation between emulation proper and the user interface allows multiple processes to control the emulator simultaneously. This is useful when controlling the emulator both interactively, and with parameters generated from previously-taken traces. New emulator settings may be loaded continuously, even to control currently-active flows. Kernel data structures are handled in such a way as to minimize the locking required. Two user interfaces are provided with the code: a simple command-line interface, suitable for scripting, and an interactive graphical interface

DRD drops packets with a probability that (after a minimum threshold is reached) increases linearly with the instantaneous queue length. DRD does have shortcomings compared to more complex router congestion control mechanisms such as Random Early Detection (RED), principally because DRD can result in coordination of packet drops and retransmissions across multiple flows after certain types of instantaneous traffic bursts. However, these shortcomings are not relevant for NISTNet, where each flow is treated separately, and hence cross-flow-coordinated drops do not occur. Based on the requirement of our application and the flexibility of the emulator we set the packet loss in three different levels. Our data is tested for 10%, 15% and 20% of data loss, while higher losses may lead to instabilities in the system which the predictor might not be able to predict the values.

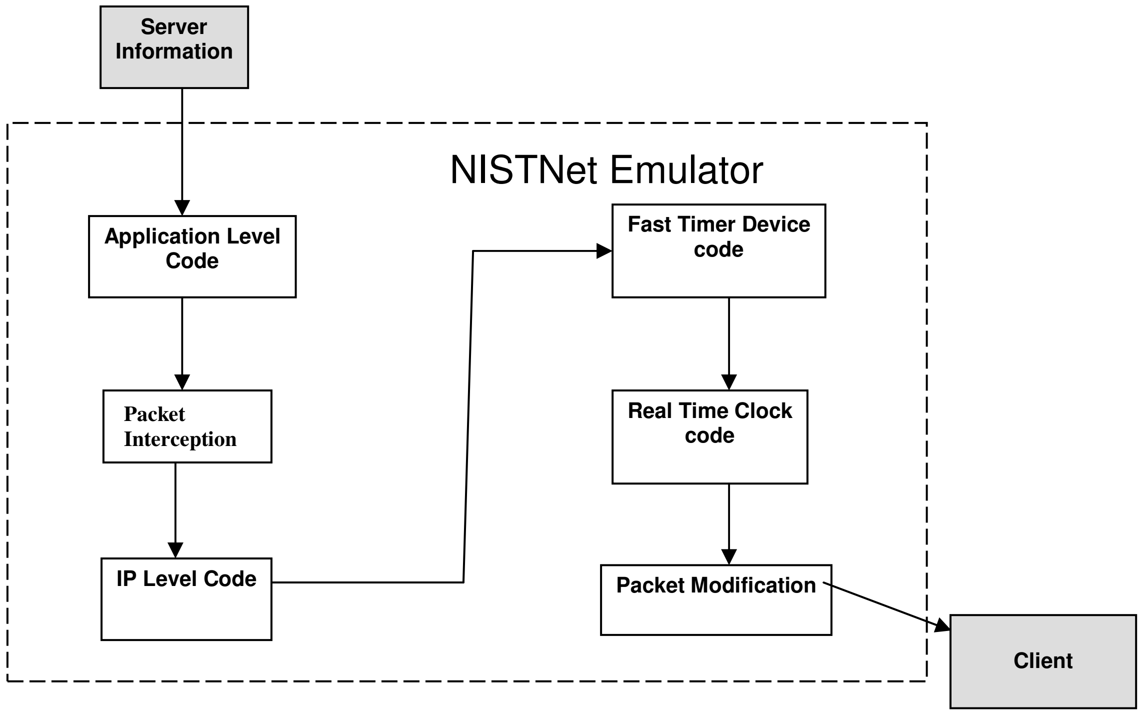


Figure 4.5 Working of NISTNet Emulator

The NISTNet kernel module makes use of two hooks into the Linux kernel. In order to inspect all incoming packets for potential handling, the *packet intercept* code seizes control of the IP packet type handler. All IP packets received by network devices are then passed directly to the NISTNet module. After *packet matching* determines (based on the table of emulator entries) whether and how *packet processing* should affect the packet, NISTNet then (possibly after delay) passes the packet on to the Linux IP level code. The *fast timer* takes control of the system real time clock and uses it as a timer source for *scheduling* delayed packets. In the process, it reprograms the clock to interrupt at a sufficiently high rate for fine-grained packet delays

4.4 CLIENT ARCHITECTURE

Client side receives the fluctuated force data packet which arrives with loss and delay. Hence if we process the data as it arrives with the modified arrival time, then it will lead to instability to the system. So applet update rate has to be adjusted as how it was generated at the source. Efficient synchronization mechanisms are required which modifies the update time at the client side so as to synchronize the data received with the server. To achieve this we propose an integrated synchronization approach which takes care of both packet loss and delay jitter induced by the network. A detailed description of the two algorithms is given in the following sections

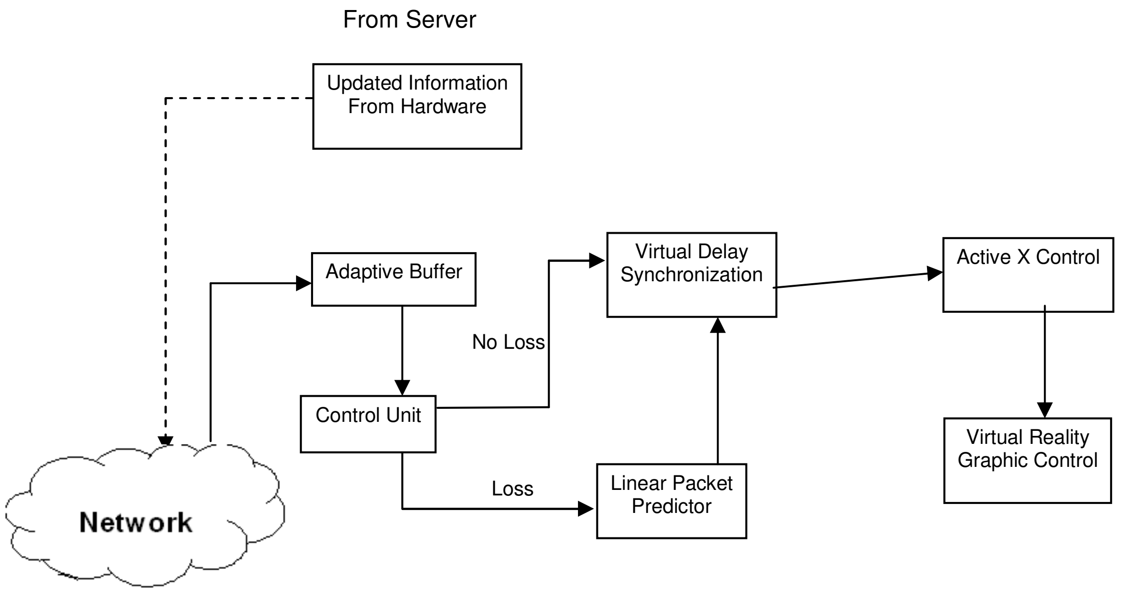


Figure 4.6 Client Architecture

Updated joystick positional and force information are received from the server side are to be used to update the applet. Data packets arrive with jitter and loss from the network and are temporarily stored in an Adaptive buffer. Data stored in the buffer are retrieved and sent to the control unit to do further processing. According to the synchronization control mechanism, a buffering process is used at the remote haptic process. The purpose of buffer is to store the arrival Data packet for delay and loss compensation. Then, the stored data packets are processed by the linear packet predictor algorithm and virtual time rendering algorithm to solve the interval fluctuation before sending to update, or display, at the client (remote side).

4.4.1 CONTROL UNIT

Control Unit receives packets from the buffer to decide which algorithm to run in order to achieve the synchronization criteria. Control Unit decides on two cases depending on the packet loss. It extracts the sequence number information from the RTP and reads to check if the packets have arrived in sequence. If so it directly sends the packet for delay adjustment to the Virtual Time Rendering algorithm. If the control unit checks for loss of data packet then it decides to transfer the packet to the Linear Packet Predictor Algorithm. Detailed explanations of the two algorithms are discussed in the next section.

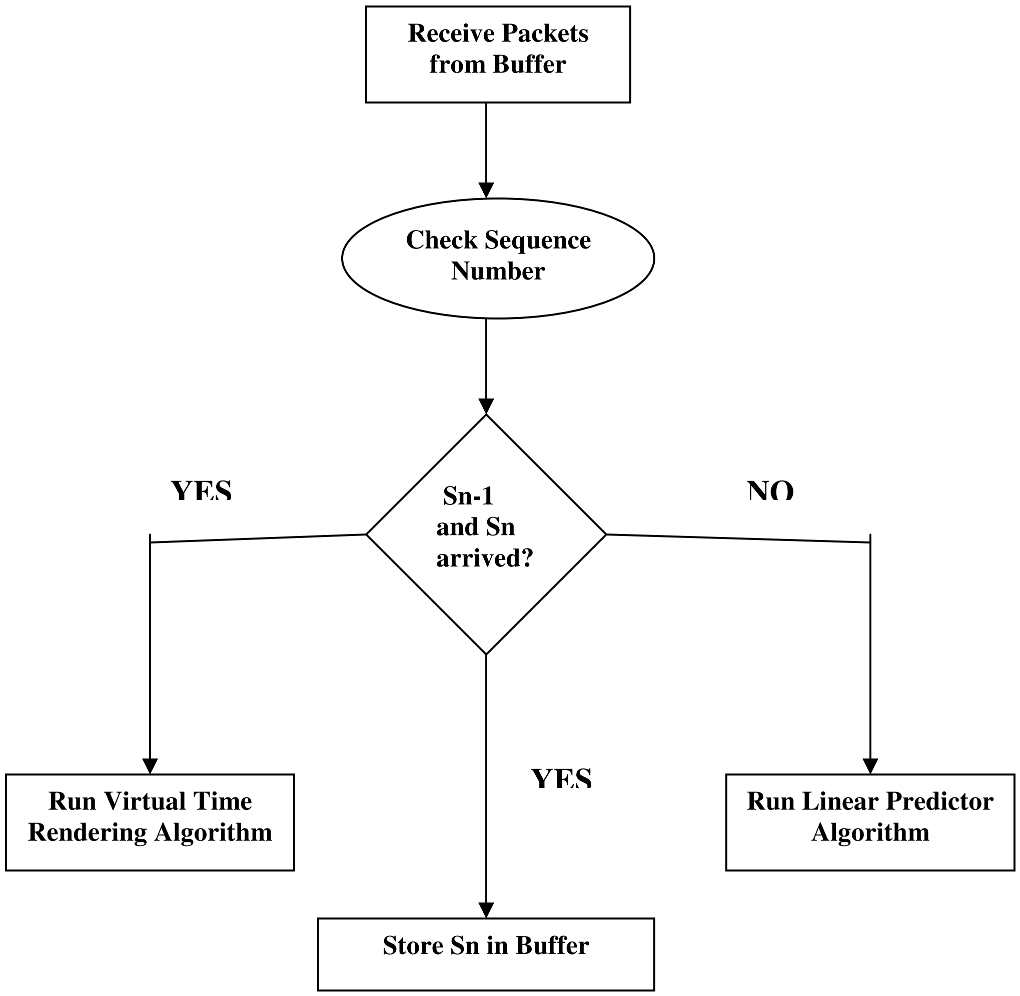

Receive Packets from Buffer
Check Sequence Number
YES
Sn-1 and Sn arrived?
NO
YES
Run Virtual Time Rendering Algorithm
Run Linear Predictor Algorithm
Store Sn in Buffer

Figure 4.7 Control Unit Decision

Data packets are received by the control unit and analyzed for the sequence number. If the packets are received in order the process is transferred to the Virtual Time Rendering Algorithm. If the data packet is lost, the process transfers to the Linear Packet Predictor Algorithm. Due to the real time nature of the application data packets cannot be retransmitted, since it adds to end to end delay ultimately reduces the efficiency of the application.

4.5 LINEAR PREDICTOR ALGORITHM

Linear prediction is a mathematical operation where future values of a discrete-time signal are estimated as a linear function of previous samples. Each time a data arrives the control unit in the receiver reads the sequence number of the arrived packet and checks whether it has arrived in sequence. If any packet is missing and out of order, then the mis-ordered packet is sent to the Linear Predictor Algorithm. The algorithm takes in the current received data packet and compares with the previously arrived packet and predicts the value.

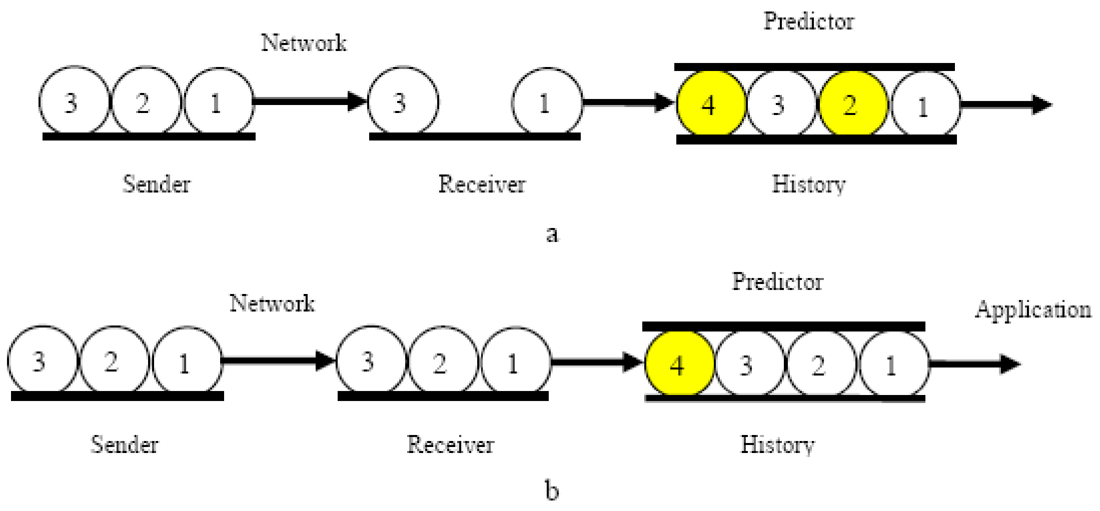


Figure 4.8: Packet Prediction a) in-between message b) last update message

4.5.1 PREDICTION MODEL

The most common representation is

$$\widehat{x}(n) = -\sum_{i=1}^{p} a_i x(n-i)$$

where $\widehat{x}(n)$ is the predicted signal value, $x(n - i)$ the previous observed values, and a_i the predictor coefficients. The error generated by this estimate is

$$e(n) = x(n) - \widehat{x}(n)$$

where $x(n)$ is the true signal value.

These equations are valid for all types of (one-dimensional) linear prediction. The differences are found in the way the parameters a_i are chosen.

For multi-dimensional signals the error metric is often defined as

$$e(n) = \|x(n) - \widehat{x}(n)\|$$

4.5.2 ESTIMATING THE PARAMETERS

The most common choice in optimization of parameters a_i is the root mean square criterion which is also called the autocorrelation criterion. In this method we minimize the expected value of the squared error E [e^2(n)], which yields the equation

$$\sum_{i=1}^{p} a_i R(i-j) = -R(j),$$

for $1 \leq j \leq p$, where R is the autocorrelation of signal x_n, defined as

$$R(i) = E\{x(n)x(n-i)\}$$

Where E is the expected value

To minimize the error we can rewrite the error as the difference in the actual value at particular time to the predicted value

$$e(n) = x(n) - \widehat{x}(n) = x(n) + \sum_{i=1}^{p} a_i x(n-i) = \sum_{i=0}^{p} a_i x(n-i)$$

4.5.3 ROOT MEAN SQUARE CRITERION

The estimation parameters are optimized based on the root mean square criterion which is also called as the autocorrelation criterion.

The rms for a collection of *n* values $\{x_1, x_2, \ldots, x_n\}$ is

$$x_{\mathrm{rms}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}}$$

The **root mean square deviation (RMSD)** (*also **root mean square error (RMSE)***) is a frequently-used measure of the differences between values predicted by a model or an estimator and the values actually observed from the thing being modeled or estimated. These individual differences are also called residuals, and the RMSD serves to aggregate them into a single measure of predictive power.

The RMSD of an estimator $\hat{\theta}$with respect to the estimated parameter θ is defined as the square root of the mean squared error:

$$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}$$

Suppose if we represent the series of actual values as one vector and the predicted values as another vector we can determine the root mean square deviation can be obtained as

$$\theta_1 = \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{1,n} \end{bmatrix} \qquad \text{and} \qquad \theta_2 = \begin{bmatrix} x_{2,1} \\ x_{2,2} \\ \vdots \\ x_{2,n} \end{bmatrix},$$

RMSD becomes:

$$\mathrm{RMSD}(\theta_1, \theta_2) = \sqrt{\mathrm{MSE}(\theta_1, \theta_2)} = \sqrt{\mathrm{E}((\theta_1 - \theta_2)^2)} = \sqrt{\frac{\sum_{i=1}^{n}(x_{1,i} - x_{2,i})^2}{n}}$$

The predicted data is sent to the Virtual Time Rendering algorithm to compensate for the delay jitter between the packets.

4.6 VIRTUAL TIME RENDERING

Three types of continuous media synchronization are required for preservation of the temporal relations. One is called *intra-stream* synchronization control, and another is *inter-stream* synchronization control. The intra-stream synchronization control is necessary for the preservation of the timing relation between data packets, each of which is the information unit' for media synchronization, such as video frames and voice packets in a single stream. The inter-stream synchronization control is required for keeping the temporal relation among plural streams. The other is referred to as *group* (i.e., inter-destination) synchronization

A number of media synchronization algorithms have been proposed for intra-stream and inter-stream synchronization to satisfy diverse requirements. The algorithms are grouped into two major classes: distributed or local. Distributed algorithms are used in network environments, while local ones are employed within a workstation need to clarify which algorithm produces the best performance among a variety of algorithms in a given environment.

$\mathbf{ts_n}$ denote the timestamp of nth data at which the source has generated at the server

$\mathbf{tr_n}$ denote the timestamp of nth data at which the data packet reached the client side

$\mathbf{td_n}$ denote the timestamp of nth data at which the data is updated at the applet

The interval of data packets at the source, arrival and update processes are denoted as

- Source Interval Time

$$Tsi = ts_{n+1} - ts_n$$

- Received Interval Time

$$Tri = tr_{n+1} - tr_n$$

- Processed Interval Time

$$Tdi = td_{n+1} - td_n$$

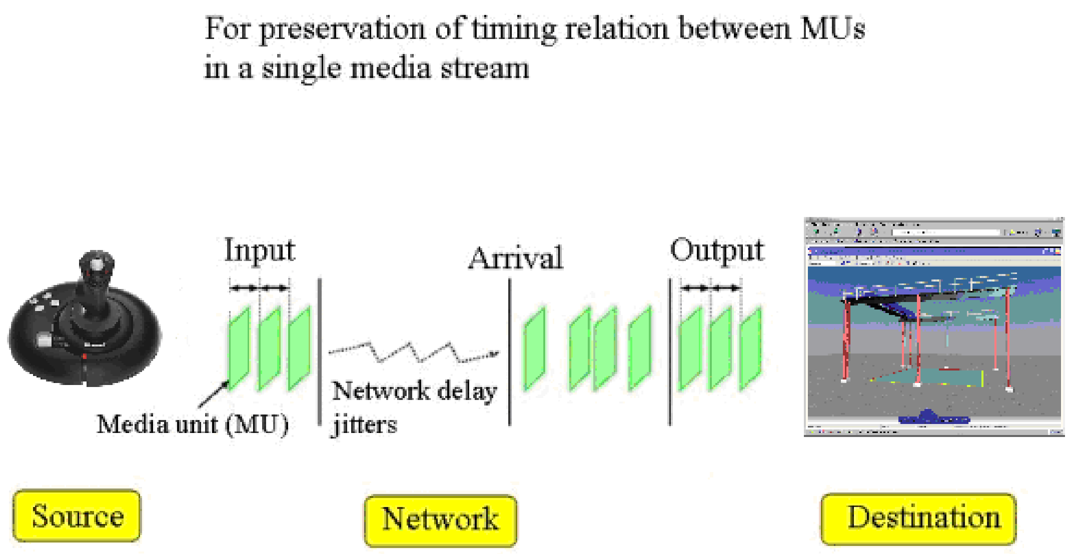


Figure 4.9 Intra Stream Synchronization

The delay variation has to be controlled. If the required received data hasn't arrived by a particular threshold time, then we can consider that the data has lost data. But if the data arrives after that we can use that data instead of the predicted data at the control unit, which requires sequentially arrived data to predict the missing packets.

The maximum delay till which the packet can wait, to decide whether to predict is given by

- Delay Variation

$$\delta = \tau max - \tau min$$

The overall end to end delay of the system determines the size of the buffer which can be set adaptively based on the delay variations.

- Delay

  $\boldsymbol{Ti = tr_n - ts_n}$

The time adjustment factor can be obtained based on the above data as

- Time Adjustment Factor
  - $\boldsymbol{\alpha i = ts_n - tr_n}$

With the algorithm calculating the time adjustment factor

We can determine the rendering rate, the update time of the applet as

- Rendering Time
  - $\boldsymbol{Td_n = ts_n + \tau i + \alpha i}$

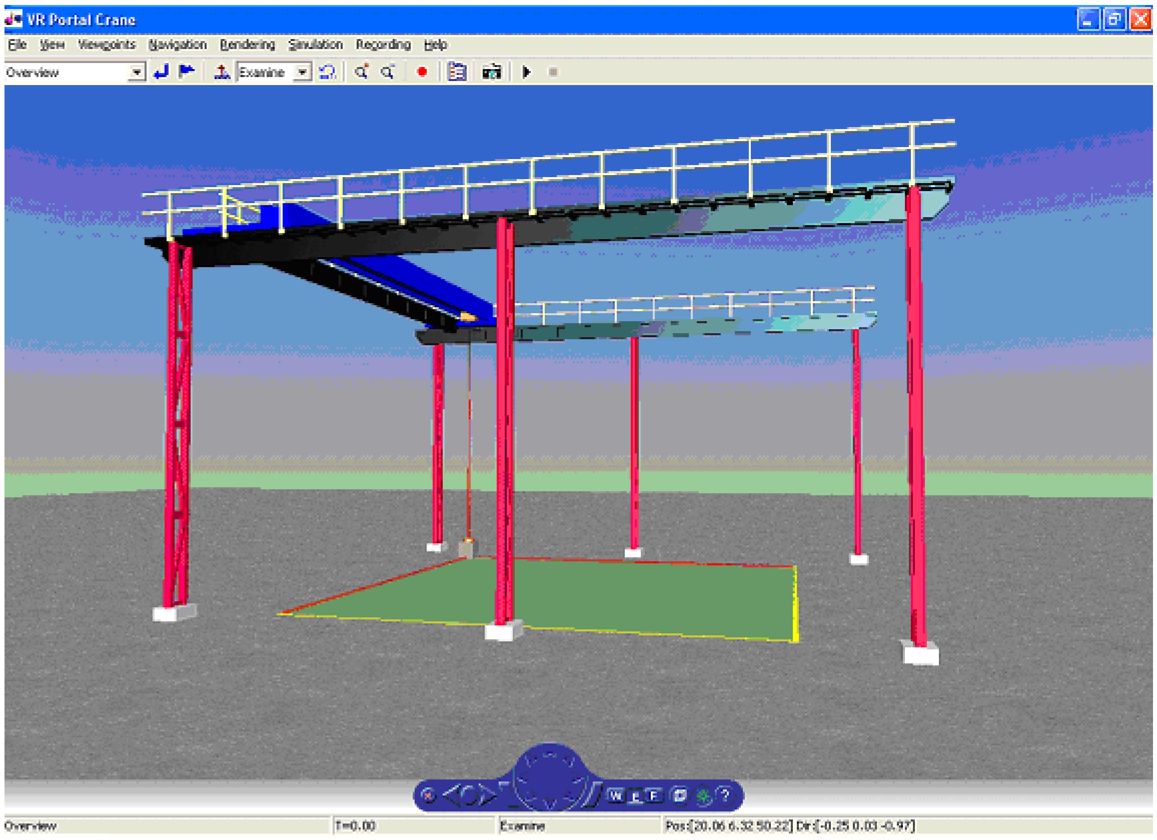
In the VTR algorithm, in addition to the actual time, we introduce a virtual-time which expands or contracts according to the amount of delay jitters of MUS received at the destination, and media are rendered along the virtual-time axis. The virtual time-expansion and time-contraction are realized in a form of modification of the target output time, which is the time when the destination should output an MU.

The application form of the modification depends on the kind of media treated, i.e., stored or live. In the case of stored media such as those in video-on-demand, the easiest way of the application is only the virtual time-expansion; that is, the target output time is postponed only. This implies that the resulting playback time becomes longer than the original recording time.

However, if the amount of the virtual time-expansion is not perceptible, the subjective quality of an output media stream can be improved. On the other hand, live media need both virtual time-expansion and time-contraction since the real-time property must be preserved. In addition to the modification of the target output time, the VTR algorithm also adopts reactive skipping and reactive pausing of MUS in order to recover from asynchrony as in many of other synchronization algorithm. Furthermore, the algorithm has functions such as preventive pausing, change of buffering time with network delay estimation and shortening and extension of output duration.

4.7 VIRTUAL REALITY APPLET

Force feedback rendering is tested based on the synchronized movement of the two devices in remote areas. When one device performs an action, corresponding movement is recorded on the remote machine. This can be tested by running an applet and avoiding a remote device. In our architecture we have developed a Virtual Reality applet which runs on the browser on the remote client side. When the source sidewinder joystick device is moved, we could control the applet movement running on the client machine.


VR Portal Crane
File View Viewpoints Navigation Rendering Simulation Recording Help
Overview
Examine

Overview | T=0.00 | Examine | Pos:[20.06 6.32 50.22] Dir:[-0.25 0.03 -0.97]

Figure 4.10 Applet at the Client Side

The applet is developed in Matlab, which contains a load hanging on the crane. The crane can move in X and Z axis. Here the Y axis movement is restricted and hence that is not taken into consideration. Initial movement of the joystick will set the place where the load will move and when the force is applied then the actual movement of the load will take place. This will result in both the position and force value transferred from the server to the client side.

The joystick generates the data and sends to the data acquisition adapter. The data is then processed and segregated into data packets. Each data packet contains both force as well as the position value. Theses UDP packets are sent through emulated network. Once the data reaches the client side, the adaptive buffer will store the data and process it to the control unit. Control Unit decides whether to run the linear predictor algorithm, if there is a packet loss, else it passes the packet to the Virtual Time Rendering Algorithm which determines when to update the applet based on the timestamp and adjusts the difference in the actual received time of consecutive data.

In the future work, we propose to build perform a set of experiment and complete the implementation of the SURGNET architecture. In the rest of the chapter, we explain different experiments we are planning to perform as future work.

4.8 **INFORMATION MULTIPLEXING:**

The SURGNET architecture supports to include various forms of multimedia information. In a general surgical unit we need to transmit the audio instructions given by the surgeon to the remote paramedic persons helping out in setting up the operational setup and audio information of the paramedic mentioning about the setup to the surgeon, video information with regular updates which are captured from the patient side so that the source surgeon side can monitor the actual position of the device. On a real application setup we need to capture the video of the patient but here we capture the updated position of the applet as a video file and transfer that to be displayed on the source side.

The applet runs on the local source machine when the device is moved, now the surgeon handling the device can compare the applet in the local stand alone side and the updated positional and force value from the applet which was received from the remote side. So we need to multiplex the information and send it. Bandwidth in a network refers to the data rate supported by the network connection. It is expressed in bits per second transferred. It represents the capacity of the connection and hence better performance is achieved with greater capacity. Certain applications require more bandwidth in order to maintain the efficient working of the application like the force feedback devices and medical data such as ECG occupies very less part of the bandwidth and hence can share the available bandwidth with such high demand applications. The SURGNET Architecture is modified when introduced with other media and can be represented as shown below.

4.9 INTRA MEDIA SYNCHRONIZATION

Intra-media synchronization control is required for keeping the temporal relationship among media streams. Lip-sync is a representative of the intra-media synchronization, and it means the synchronization between spoken voice and the movement of the speaker's lips. Media streams fall into a master stream and slave streams. Since we have two multimedia multiplexed in our application we maintain the higher bandwidth requirement media as the master stream. So force feedback data streams act as the master streams and we have video as slave stream which is required to be synchronized with the master stream.

For example in lip-sync, the voice is generally selected as the master stream. This is because voice is more sensitive to intra-stream synchronization error than video. The rendering time among various media should be modified to based on the synchronization requirement. The Intra media Synchronization can be described as below.

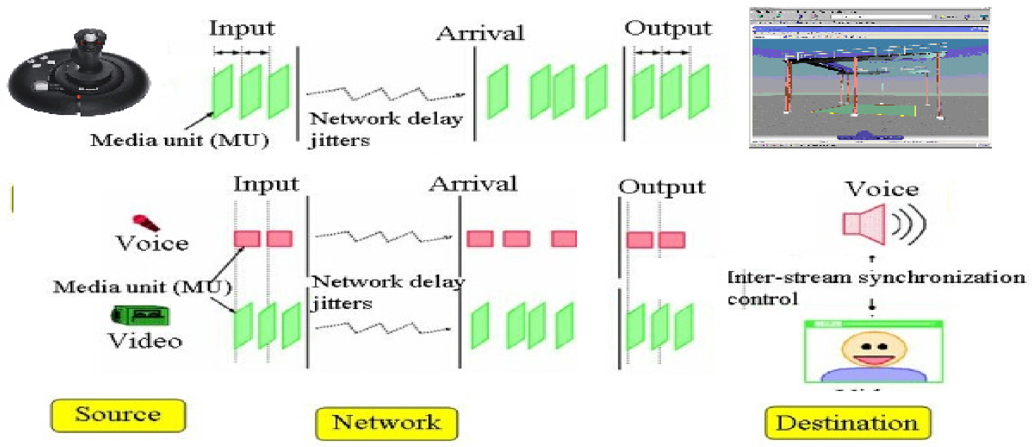


Figure 4.11 Intra Media Synchronization

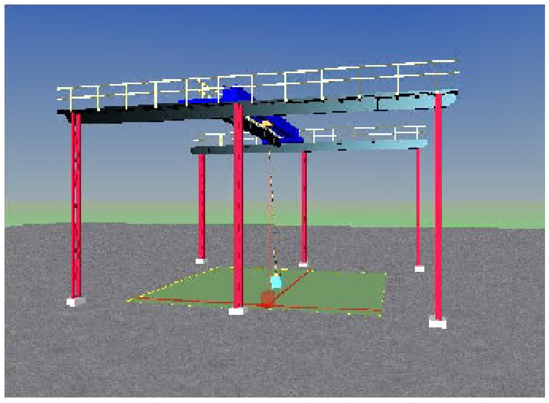
4.10 IMAGE ANALYSIS AT THE SERVER SIDE

Feedback is sent to the server by capturing the current movement of the applet, recorded as video at 30 frames/sec and streamed back. The server side runs various Image analysis tools which can select to receive the continuous packets by using the monitor mode or rather switch to playback mode to stop receiving current packets and perform Image analysis over the received images.

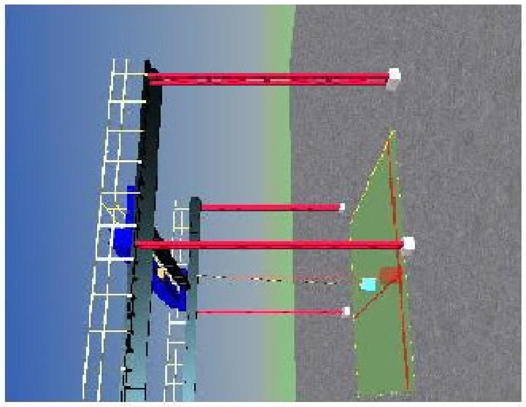
Image analysis at the server side involves contrast, brightness modifications, gray scale image generation, image rotation, image segmentation and zooming of the images. Each analysis is helpful in understanding more about the particular applet running on the client side.

Image Rotation

Image rotation is transforming the image in either clockwise or anti-clockwise direction. The rotation is calculated by computing the inverse transformation for every pixel in an image. Output pixels are calculated using bilinear interpolation of the original image. The colored image is computed by evaluating one color plane at a time. Once each color plane is transformed the final image with the rotated angle is obtained. The original image can be rotated in 90,180, 270 degrees in both clockwise and anti-clockwise direction. The original image is of MxN pixels with the image representation computed with a linear scaling on the pixel. Image is a unit8 indexed or RGB array rotates the image array and returns it in the output image. Figure 4.12 shows the original image and the anti-clockwise rotated flip sided image



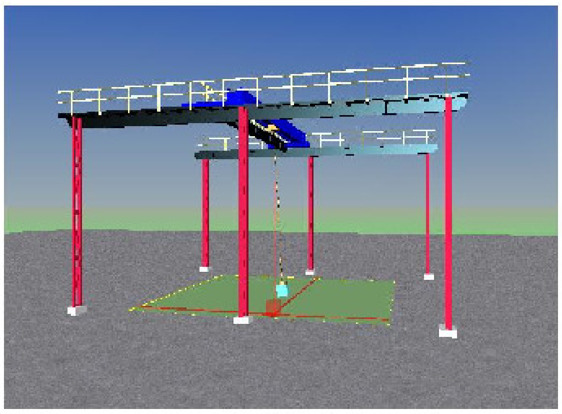
4.12.1 Original Image

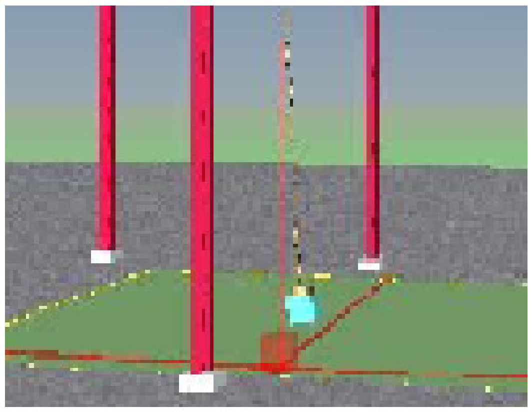


4.12.2 Rotated/Flipped Image

Image Segmentation

Image Segmentation performs partitioning of image into multiple regions. This is significantly useful when a particular region is of interest for further processing. The result of segmentation is set of regions that together form the entire image. Adjacent regions are significantly different with respect to the same characteristic. We use built in methodology using clustering methods to perform the segmentation of image with region of interest. Figure 4.13 shows the segmented image with region of interest focusing the movement of the crane on the applet

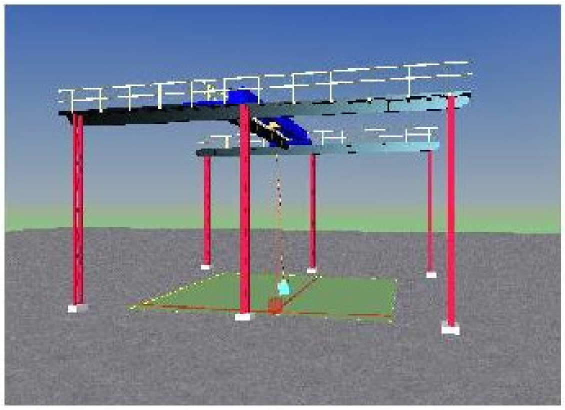

4.13.1 Original Image

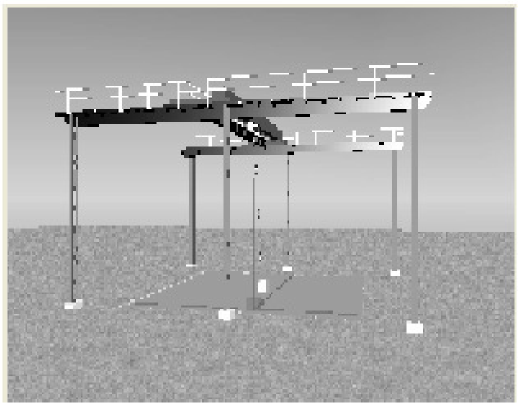

4.13.2 Segmented Image

Gray Scale Image Generation

A gray scale image is one in which the colors are shades of gray. This will require less information to be provided to each pixel. The gray color is one in which red, green and blue components have equal intensity in RGB space with single intensity value of each pixel. Gray scale image is used for further processing to get additional information of the original data. To convert a color image to its most approximate level of gray, we need to obtain the values of red, green and blue primaries in linear intensity. With this as input we add 30% of red value, 59% of the green value and 11% of the blue value together.

The resultant number is the desired gray value. The percentages are due to the different relative sensitivity of the human eye to each of the primary colors. The final result is the linear intensity value of the equivalent gray which can be gamma corrected. Figure 4.14 shows the gray scale generated image from the original image


4.14.1 Original Image


4.14.2 Gray scaled Image

The primary goal of introducing video data to the application is to improve the collaborative work of the application and to determine the impact of the video data on the synchronization of the force feedback device and the applet on the client side. In our work we have not concentrated in the bandwidth issues of the application as we deal with a simulated single hop network. Our main concern is to determine the impact of the video data on the synchronization of the force feedback device with the applet in the client side. We have developed the Therapist GUI at the server which will control all the above image processing tools during the playback time. Figure 4.14 shows the therapist GUI to control the Image processing application and based on that the future movement of the device is controlled.

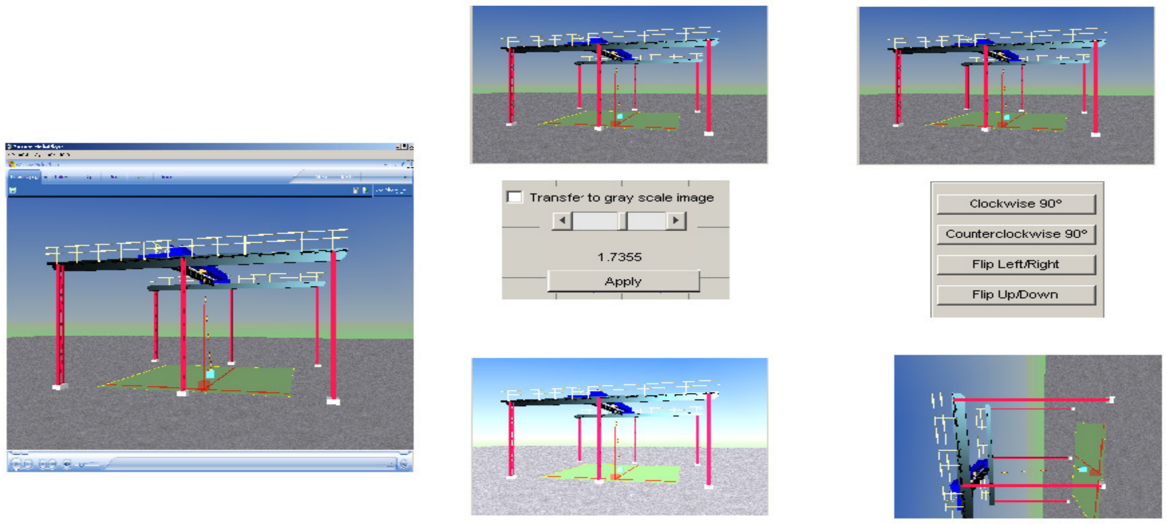


Figure 4.15 Therapist GUI

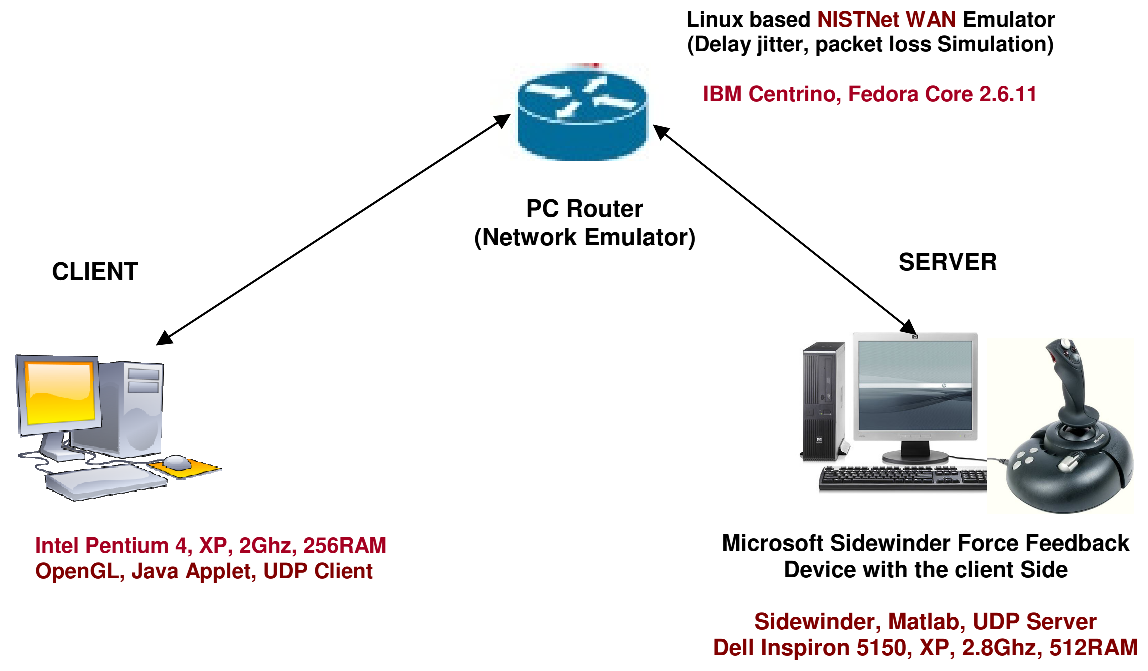
The above figure shows that the real time video is streamed from the client side and captured through a media player. The server has the option of operating in two different modes. During the monitoring mode the data is played in the player. When the therapist decides to perform Image processing analysis on the received data, the video capture is stopped and current update from the client is loaded on the player. With this as input, various image processing techniques can be performed on the received image as discussed in the previous sections. In our results section we will determine the impact of the video on the force feedback data.

CHAPTER 5

PRELIMINARY RESULTS

5.1 TEST BED

The test bed shows the setup of Client Server architecture. Here we have the Microsoft Sidewinder Force Feedback Device attached with the Server. The Server runs the Matlab Processing Unit which takes input from the joystick and reads the Axis and button values from the data acquisition Adapter and runs the UDP Server. The Server runs on a Dell Inspiron 5150 on Windows XP machine, with 2.8 GHz and memory of 512RAM.



Linux based NISTNet WAN Emulator
(Delay jitter, packet loss Simulation)
IBM Centrino, Fedora Core 2.6.11
PC Router
(Network Emulator)
CLIENT
SERVER
Intel Pentium 4, XP, 2Ghz, 256RAM
OpenGL, Java Applet, UDP Client
Microsoft Sidewinder Force Feedback
Device with the client Side
Sidewinder, Matlab, UDP Server
Dell Inspiron 5150, XP, 2.8Ghz, 512RAM

Figure 5.1 Test Bed for SURGNET

The Network Emulator (NISTNet) is setup on a IBM Centrino, Linux machine. The machine is treated as a router which takes input from the server, process and updates the client side. NISTNet Emulator runs on a Fedora Core 2.6.11 machine with Real Time Clock setup as separate module from the Kernel.

The Client side runs the Virtual Reality (Applet) enabled Web browser. The Client side runs the UDP Client which runs on an Intel Pentium 4, XP with 2 GHz and 256 of RAM Memory. Unlike the TCP connection where the Server has to be starter first, here the UDP Client and Server can be started independently.

5.2 RESULTS

The generated data is recorded at the source which shows the movement of the joystick at the server side. For convenience we have recorded just the X-axis movement. In the below generated data we see that the X-Axis position moves in the negative axis also. That is because of the base of the handle in the Sidewinder Joystick is located in the center. So we have programmed in such a way that if the handle moves to the left it gives a negative value and when it moves to the right it generates a positive value in the X-Axis position.
Since Force feedback rendering rate is 1 KHz we see that the data is generate every 1msec. Hence immediate movements of the handle of the joystick are recorded for first 50msec. A random movement was recorded and plotted in MatLab which stored the readings in a mat file.

In order to obtain efficient synchronization mechanisms, integrated synchronization algorithms ensure that both the packet loss and the delay jitter are compensated. In our study we will determine the output of the algorithms separately and show why it is necessary to have an integrated compensation technique so that Synchronization is achieved.

Our analysis involve showing the output of data loss separately and the delay jitter separately, and insisting on the combined effect. Later we will show the impact of video data on the synchronization of force feedback device. Our Synchronization algorithm shows that it is independent of the introduction of video data. In our experiment we have recorded the data generated by moving the force feedback device. As mentioned earlier since the joystick handle is located at the center of the base the positional data is recorded in both positive and negative x- axis. For simplicity we have recorded only the position of the x-axis and did not consider the y-axis position which will also yield similar results.

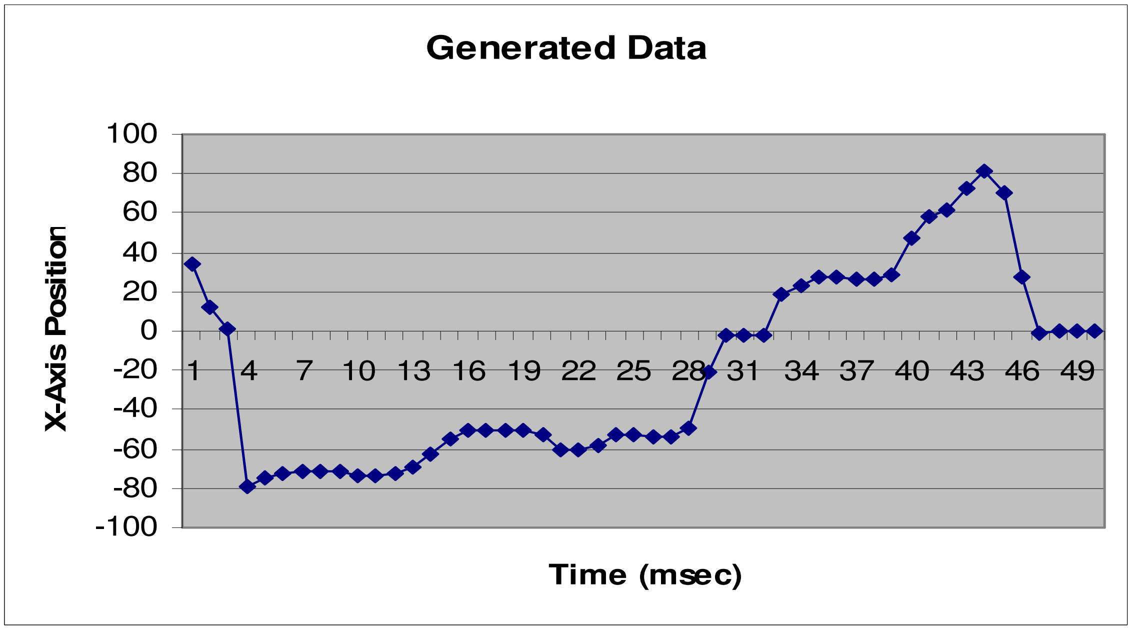


Figure 5.2 Generated Data at the Source

As seen in figure 5.2, the generated data was recorded due to the random movement of the joystick handle and that positional data variations are within the maximum limit of 100 in the positive axis and -100 in the negative axis. Any data which tries to exceed the maximum value will be rounded to the maximum value. This is not a concern in our research as we are not dealing with the precision of the device.

By introducing emulated conditions with NISTNet, we introduce a random delay and 10%, 15% and 20 % of packet loss. This random nature of packet loss is introduced in between packets. The results are evaluated without the video data and with video data to determine the impact of video on the synchronization of the force feedback data.

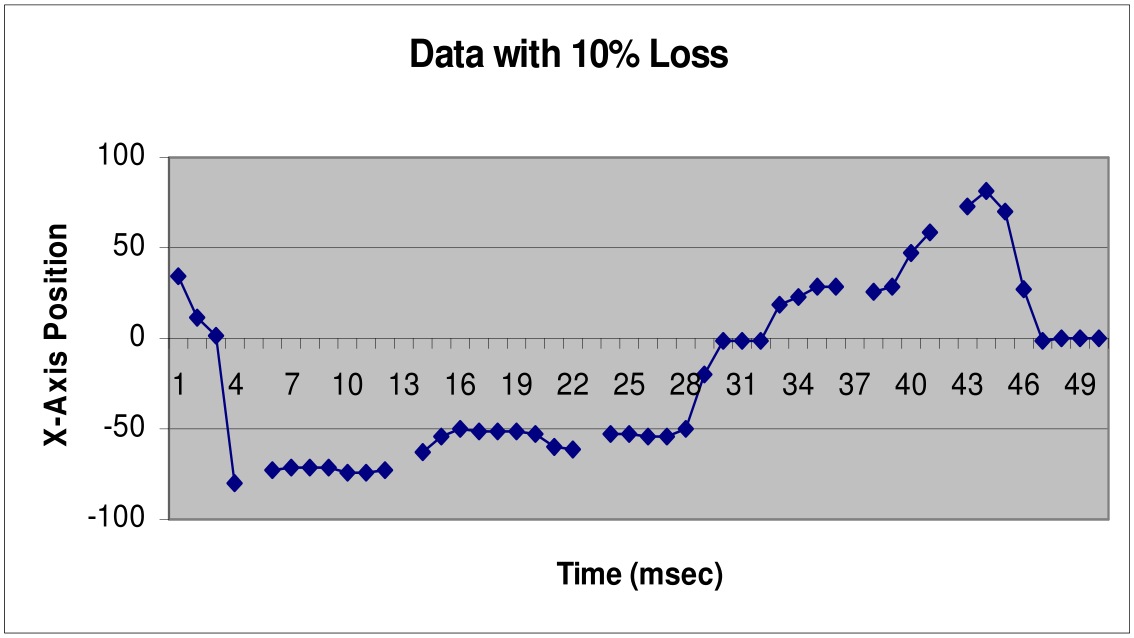


Figure 5.3 Data Loss of 10% without Video

Figure 5.3 shows the loss of packets in a random manner with 10% loss. So continuity of the data is lost. This graph does not include any delay jitter between the packets and hence just contain the packet loss scenario.

(Predicted value showing the first 15msec readings)

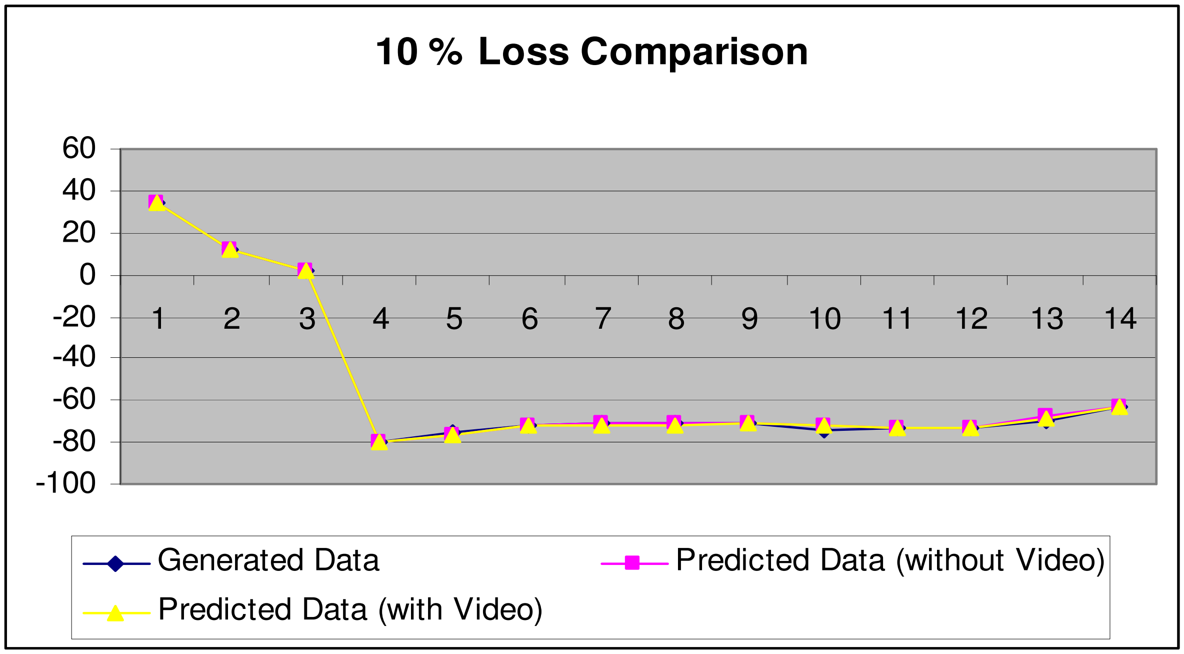


Figure 5.4 Data Comparison with Predicted (10 % Loss)

Generated Value at Source	Predicted Value at Receiver (Without Video)	Predicted Value at Receiver (Without Video)
-74.9476	**-76.0639**	-77.06
-73.69	**-72.6148**	-72.32
-69.32	**-69.477**	-68.4

Table 5.1 Data Comparison with Predicted (10% Loss)

In table 5.1 we see that the Linear Packet Predictor Algorithm was able to recover the lost packet to as close as possible. The predicted data is shown without and with video data introduced. The data predicted with video shows varying packet predicted value than predicted value without video. We had a 10 % of packet loss and hence we see that the predictor works well.

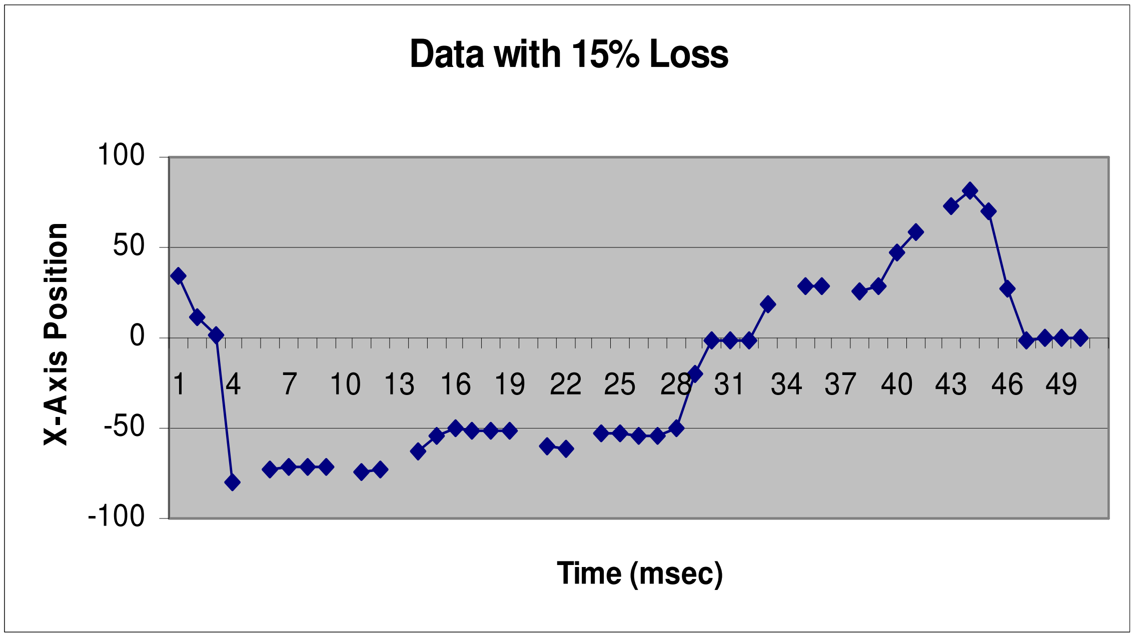


Figure 5.5 Data Loss with 15% without Video

In figure 5.5 we have recorded a packet loss rate of 15%. Here we are discussing the packet loss as a separate phenomenon and hence use only the Linear Packet Predictor Algorithm but not the VTR algorithm because there is no delay involved in this emulated condition.

(Predicted value showing the first 15msec readings)

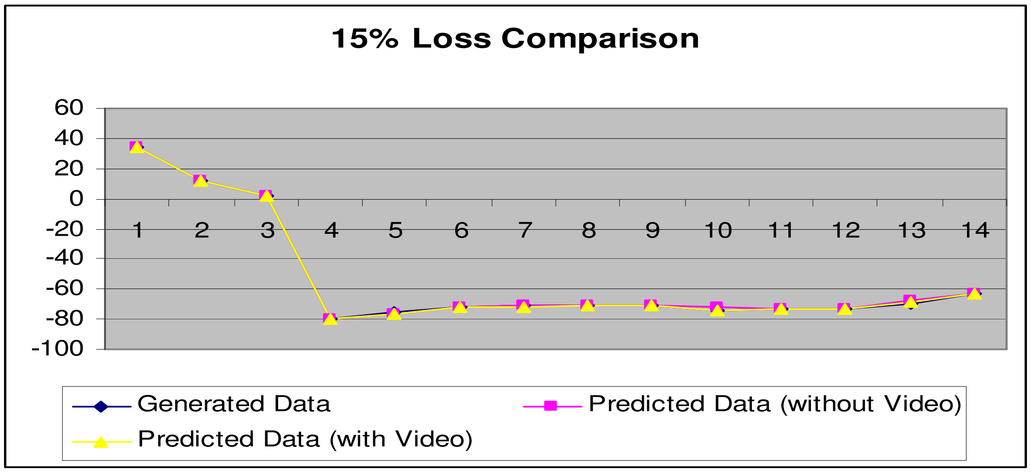


Figure 5.6 Data Comparison with Predicted (15 % Loss)

Generated Value at Source	Predicted Value at Receiver (Without Video)	Predicted Value at Receiver (Without Video)
-74.9476	**-76.606**	-76.68
-73.69	**-72.36**	-73.92
-69.32	**-67.9**	-68.98

Table 5.2 Data Comparison with Predicted (15% Loss)

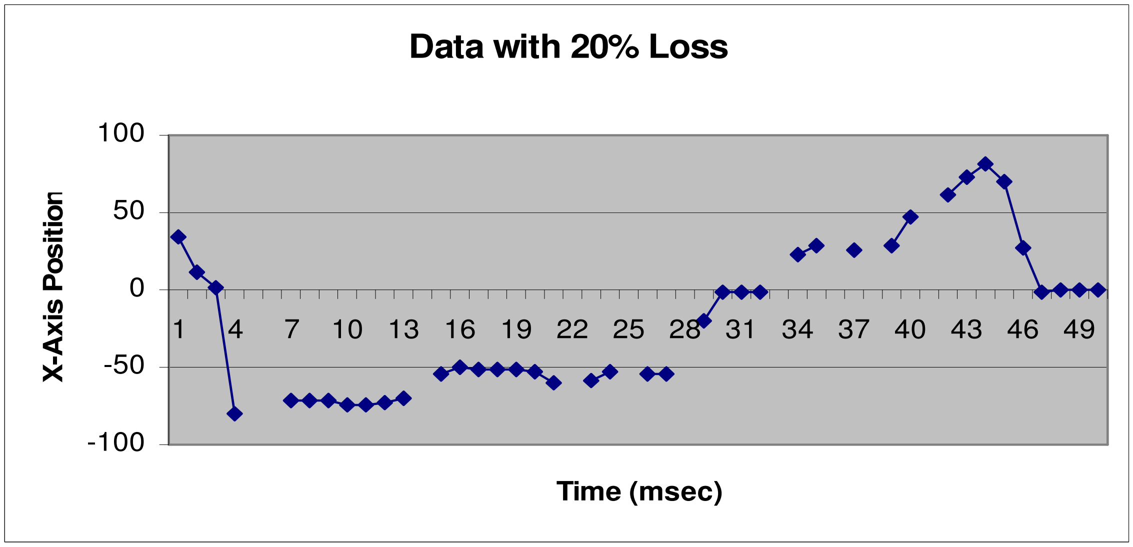


Figure 5.7 Data Loss of (20%) without Video

(Predicted value showing the first 15msec readings)

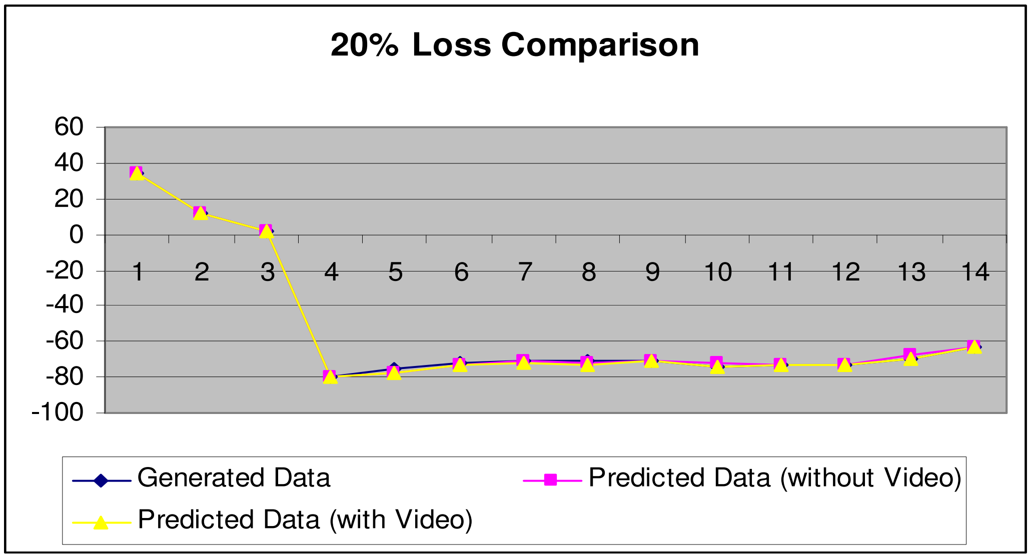


Figure 5.8 Data Comparison with Predicted (20% Loss)

Generated Value at Source	Predicted Value at Receiver (Without Video)	Predicted Value at Receiver (Without Video)
-74.9476	**-77.2638**	-77.12
-73.69	**-73.121**	-74.12
-69.32	**-72.1**	-69.4

Table 5.3 Data Comparison with Predicted (20% Loss)

From the figure we see that as the packet loss rate increases the number of packets lost increases. This factor affects the predictor when adjacent packets are lost, it would have a packet predicted from a predicted packet and hence while the loss is less the Packet Predictor predicts as close to the actual value of the packet. A comparative table below shows for a single packet which was lost on all three conditions and shows the predicted value which varies for the same packet.

Data Comparison for various losses on a single packet

Generated Data	Predicted Value (10% Loss)		Predicted Value (15% Loss)		Predicted Value (20% Loss)	
	(No Video)	(Video)	(No Video)	(Video)	(No Video)	(Video)
-74.9476	-76.0639	-77.06	-76.606	-76.68	-77.2638	-77.12

Table 5.4 Various Losses for single packet

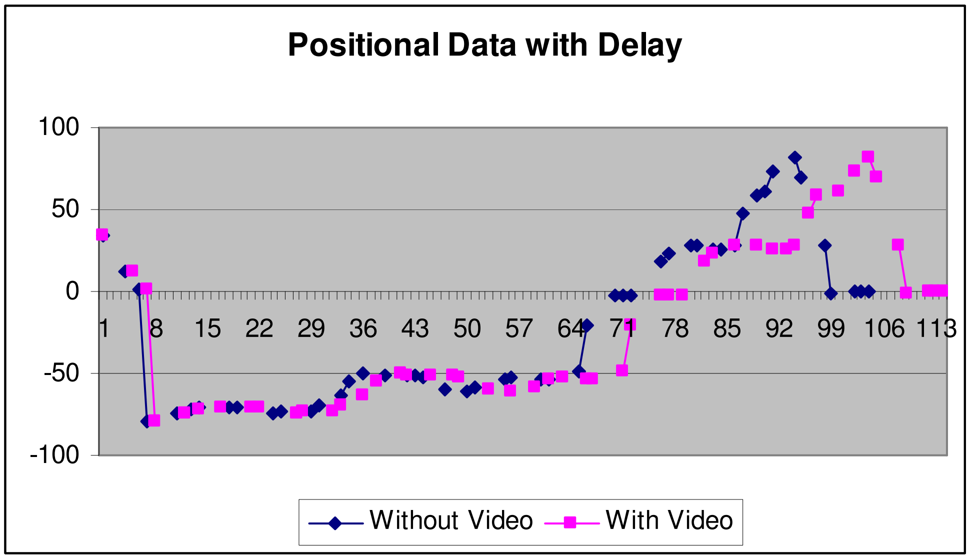


Figure 5.9 Data with variable Delay

In this condition the network will impose variable delay with heavy tail distribution between packets which reach the destination in sequence but not in the speculated time. Network delay is increased when introduced with video, but our algorithm still able to recovered the update time of the packets. We do not see any packet loss here and if we apply the Virtual Time Rendering Algorithm to the delayed data, then we can get back the required packets in synchronization.

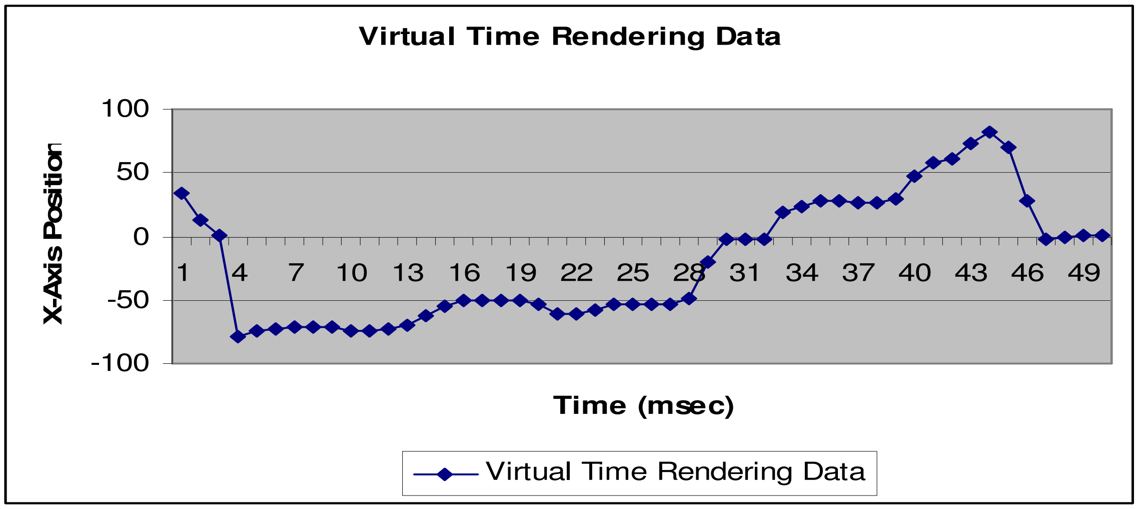


Figure 5.10 Virtual Time Rendering Algorithm

As shown in the above conditions the loss and delay does not appear separately in the network. The network imposes a combined loss and delay jitter and hence we need to compensate for both the issues at the same time. Following graphs show that a combination of variable delay with 10%, 15% and 20% loss and we will examine a step by step solution how the control unit decides to which algorithm runs first to achieve the synchronization. First we examine the 10 % packet loss with variable delay

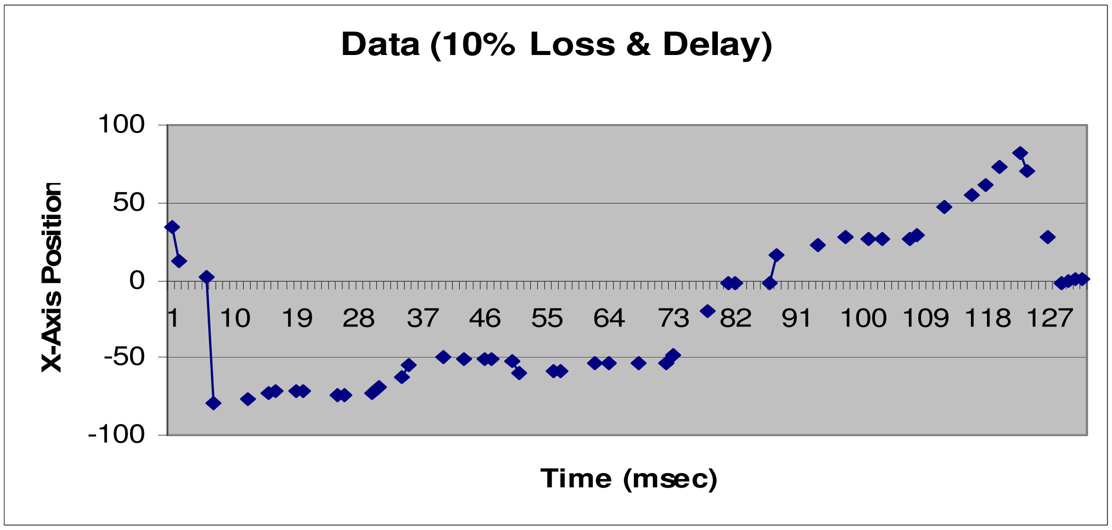


Figure 5.11 Integrated - 10 % Loss and Variable Delay

The control unit obtains the data from the adaptive buffer and checks the sequence number of the packets, if they have arrived in sequence then the data packet is passed to the Virtual Time Rendering Algorithm. If the data packet is lost then the data packet is passed to the Linear Predictor Algorithm which predicts the closest value of the packet and sends it to the VTR to compensate for the delay.

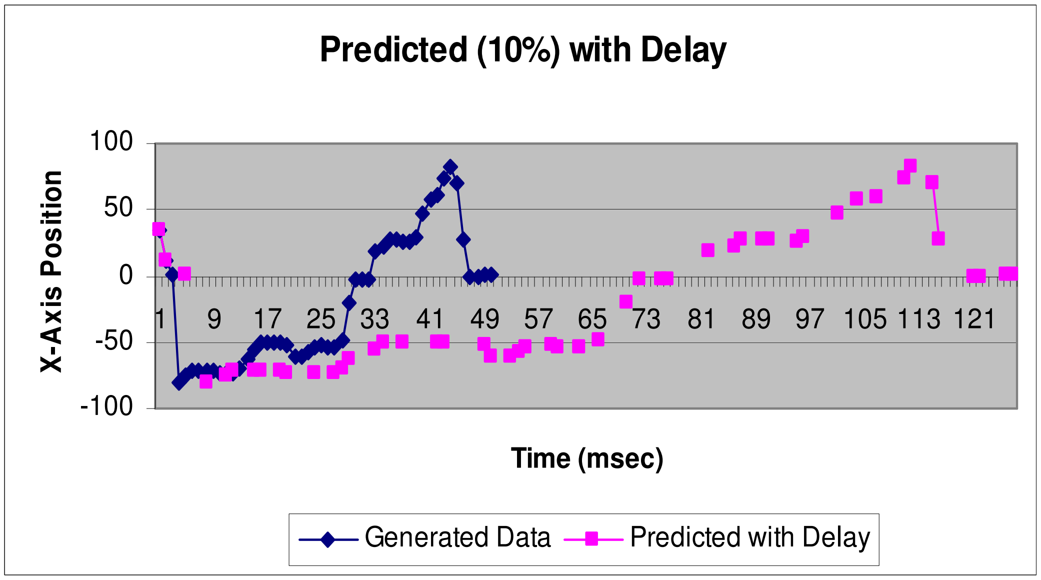


Figure 5.12 Data Comparison after Linear Predictor without Video

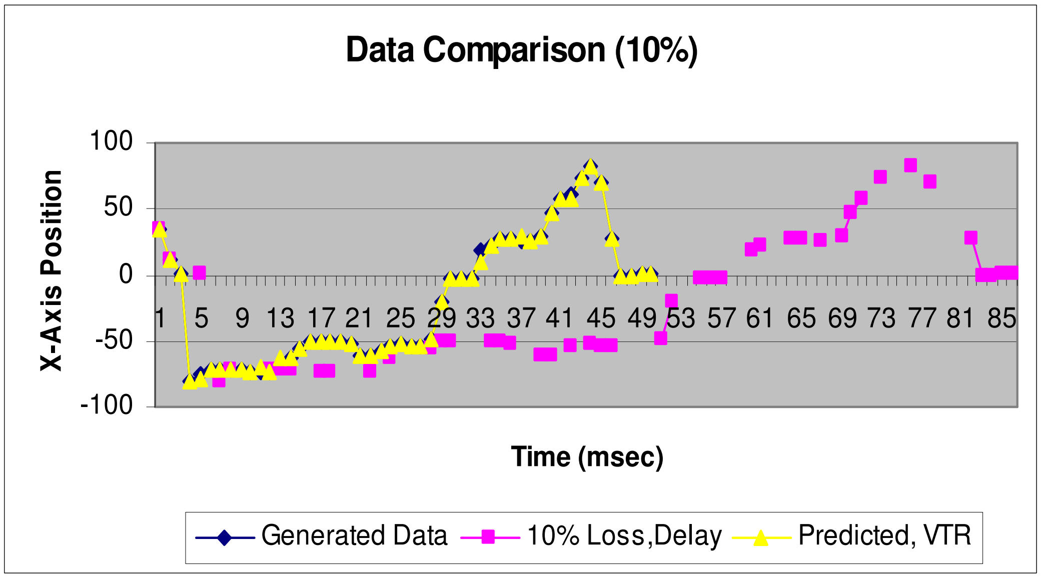


Figure 5.13 Data Comparison after Linear Predictor and VTR

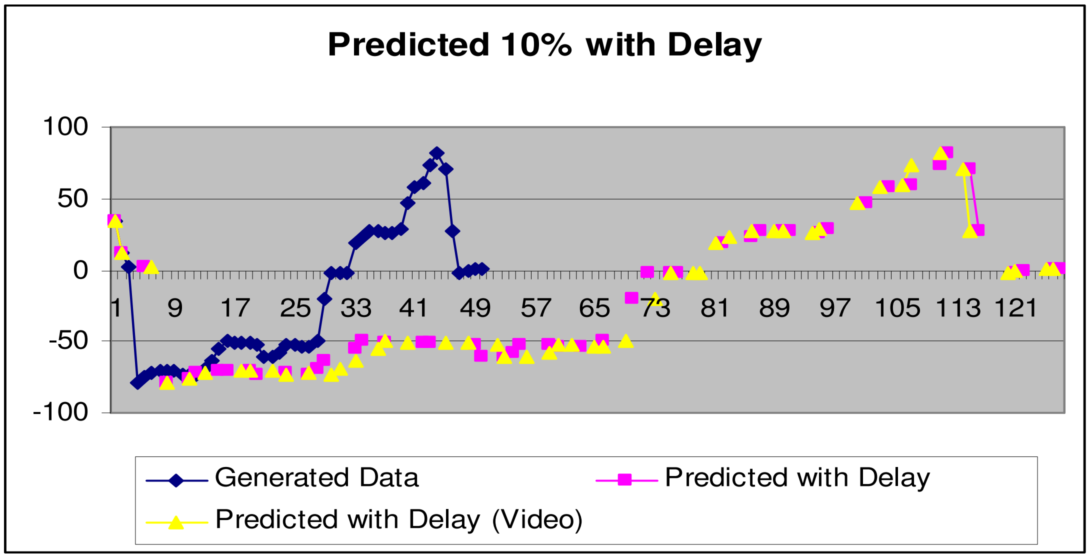


Figure 5.14 Data Comparison after Linear Predictor with Video

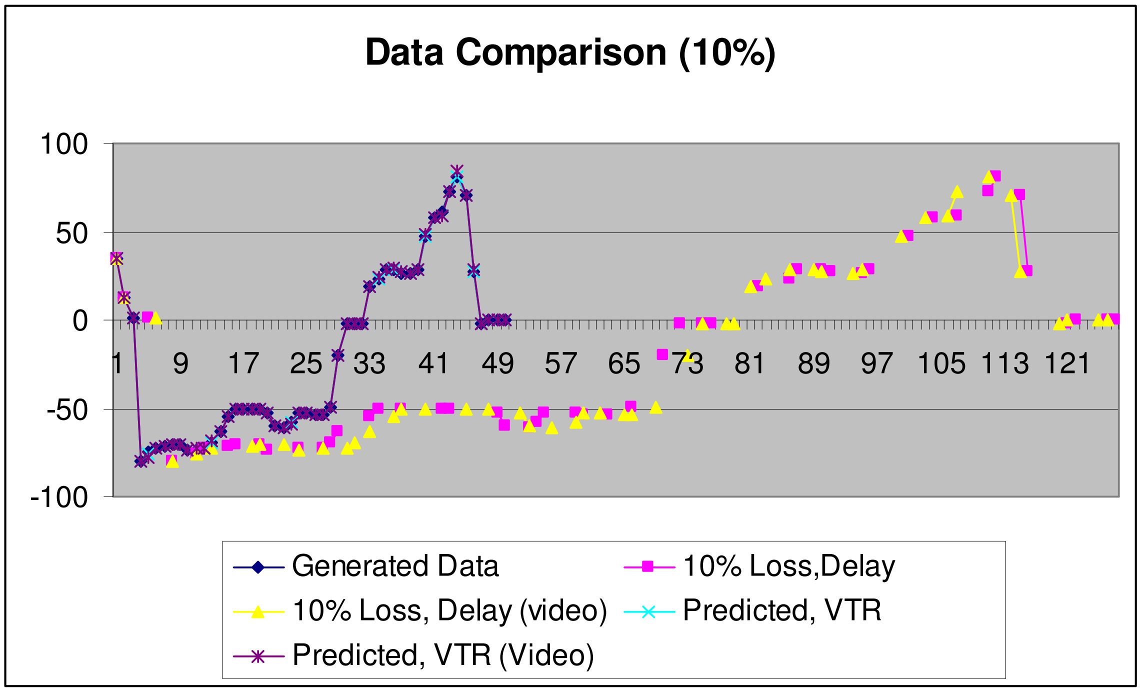


Figure 5.15 Data Comparison after Linear Predictor and VTR (Video)

Similarly we determine the output after the Linear Predictor and then the output which shows after the VTR is applied without video and with video for 15 %, 20% Packet Loss

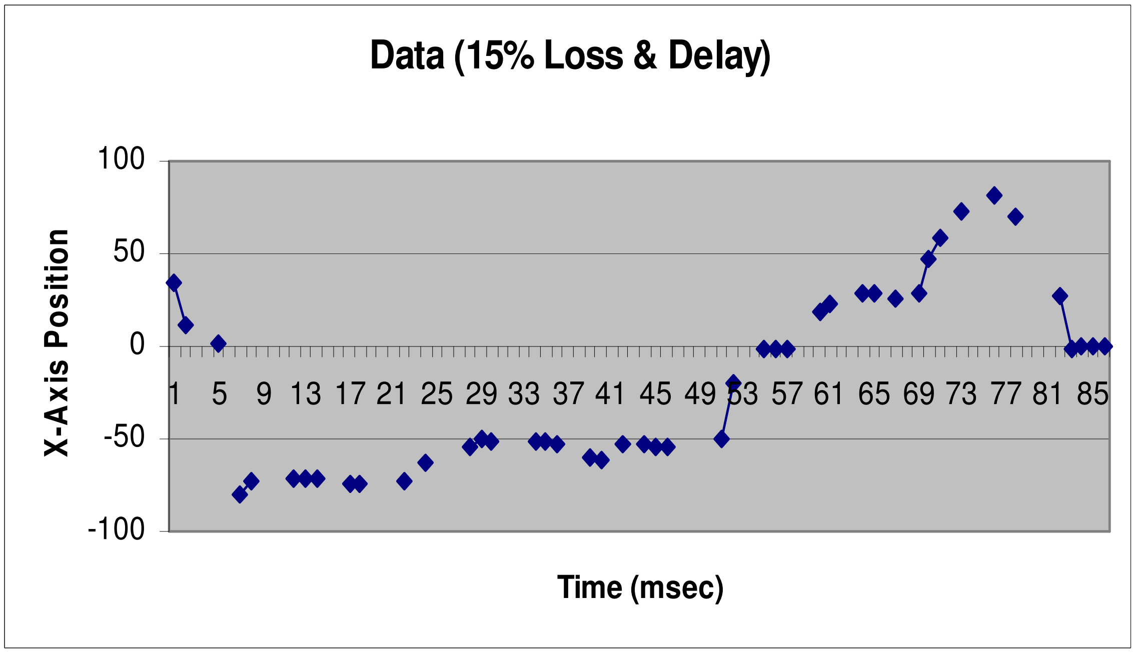


Figure 5.16 Data Comparison with variable delay and 15% Loss

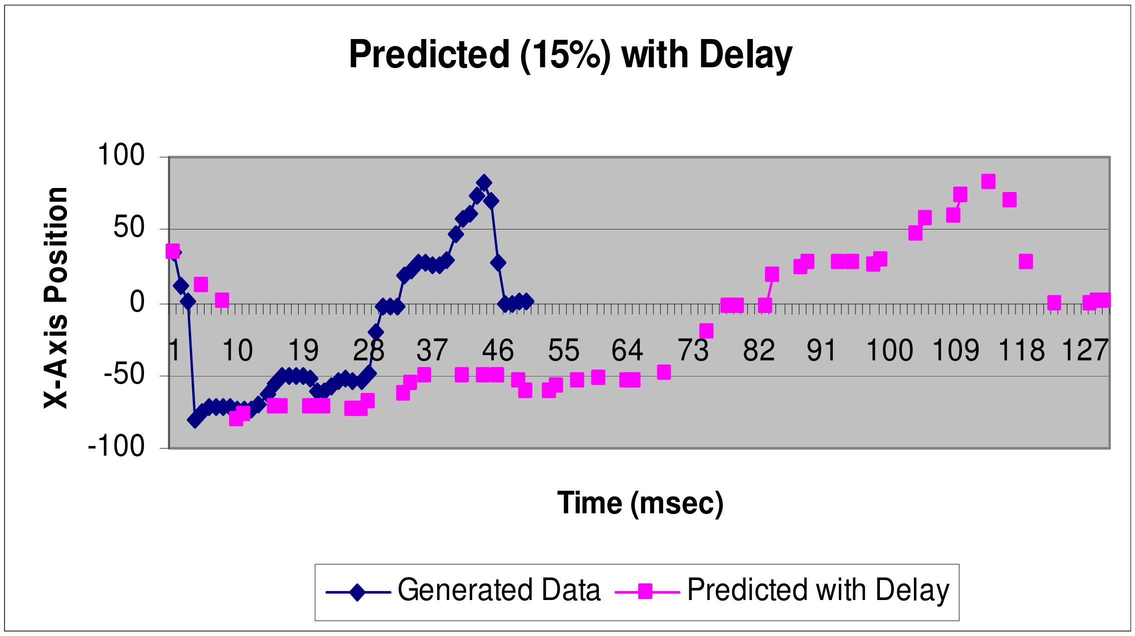


Figure 5.17 Data Comparison after Linear Prediction Algorithm

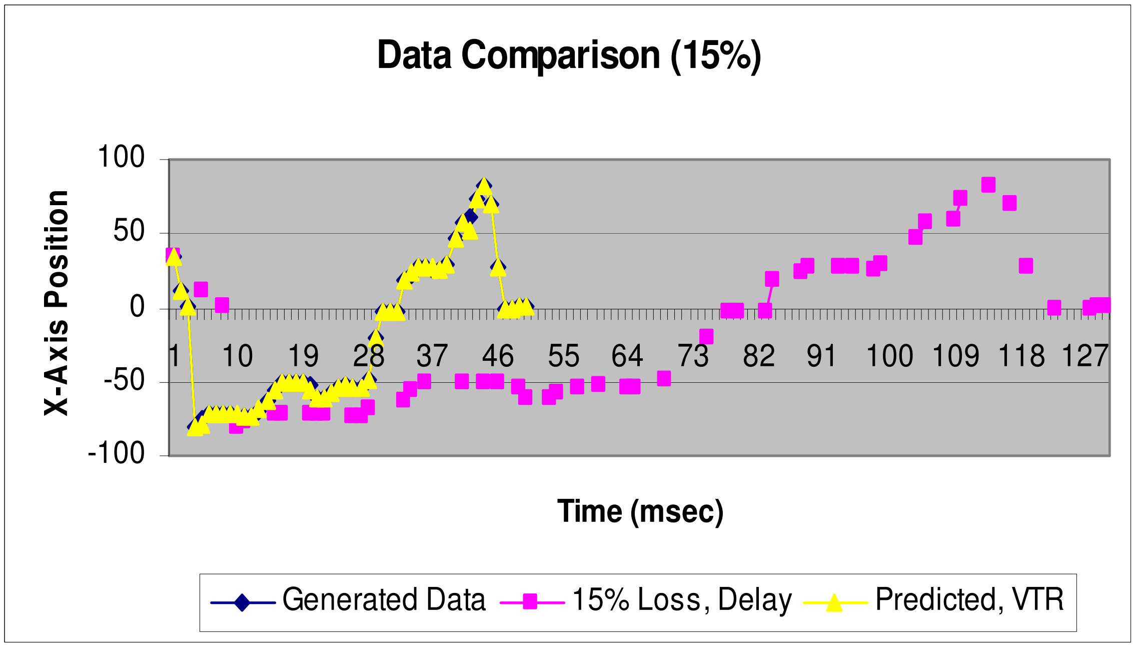


Figure 5.18 Data Comparison after Linear Predictor and VTR

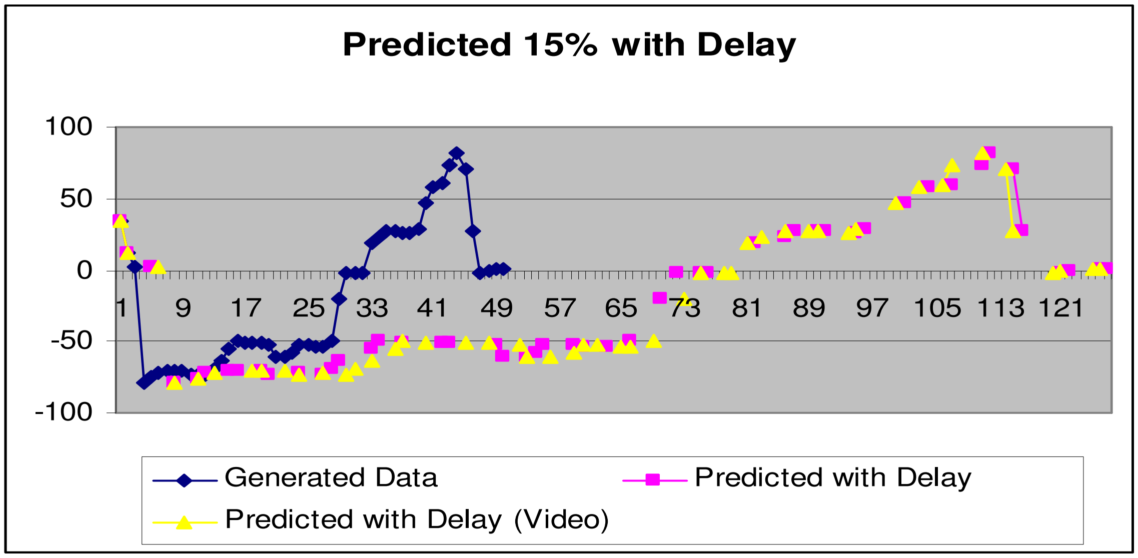


Figure 5.19 Data Comparison after Linear Predictor with Video

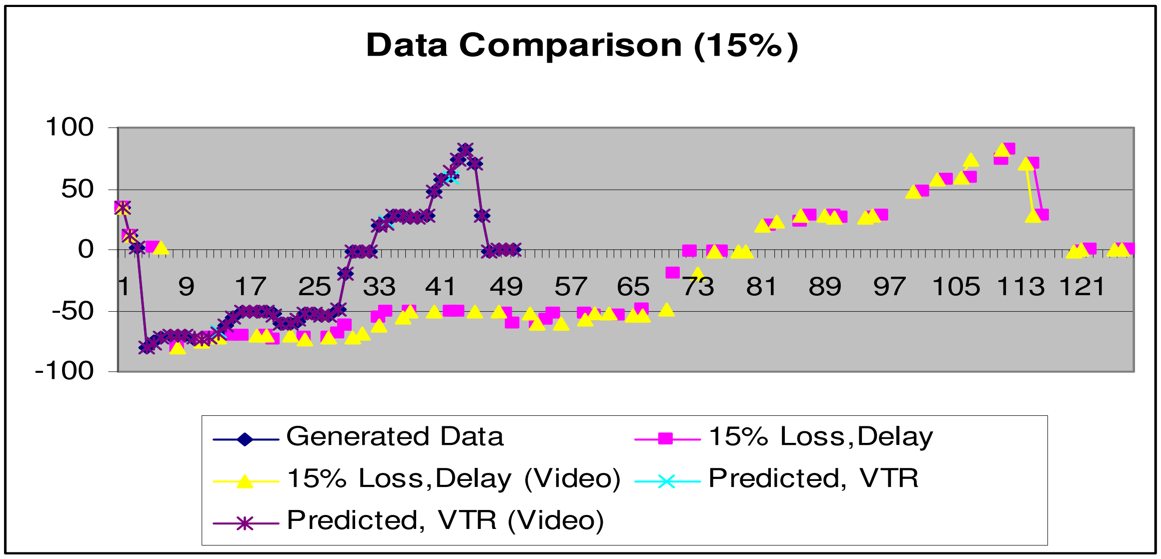


Figure 5.20 Data Comparison after Linear Predictor and VTR (Video)

Data Comparison for 20 % Loss

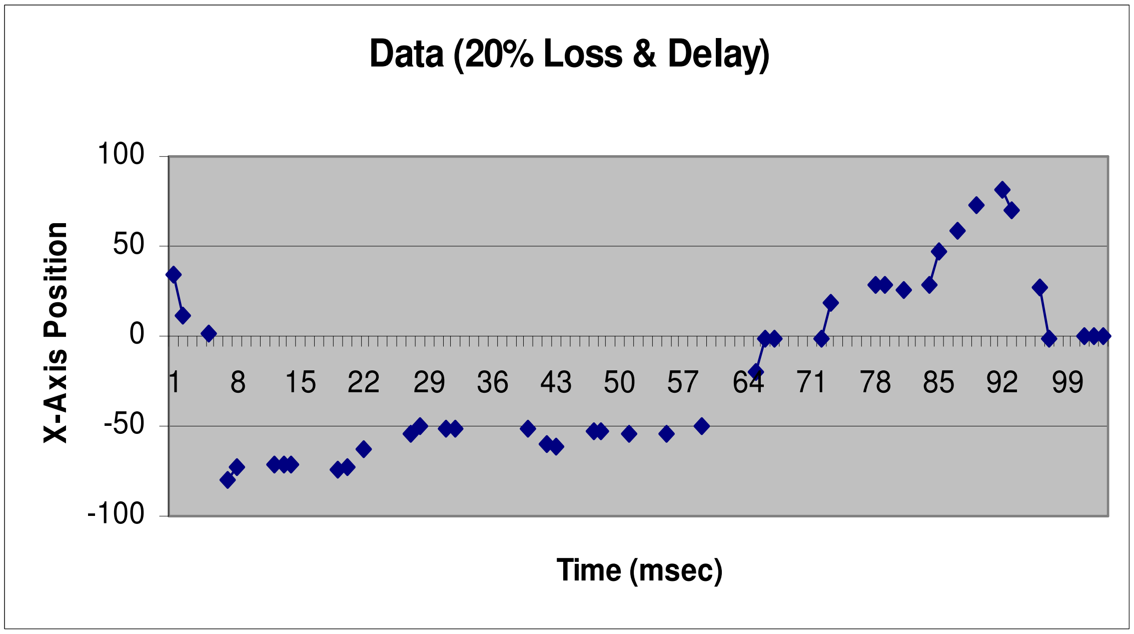


Figure 5.21 Data Comparison with variable delay and 20 % Loss

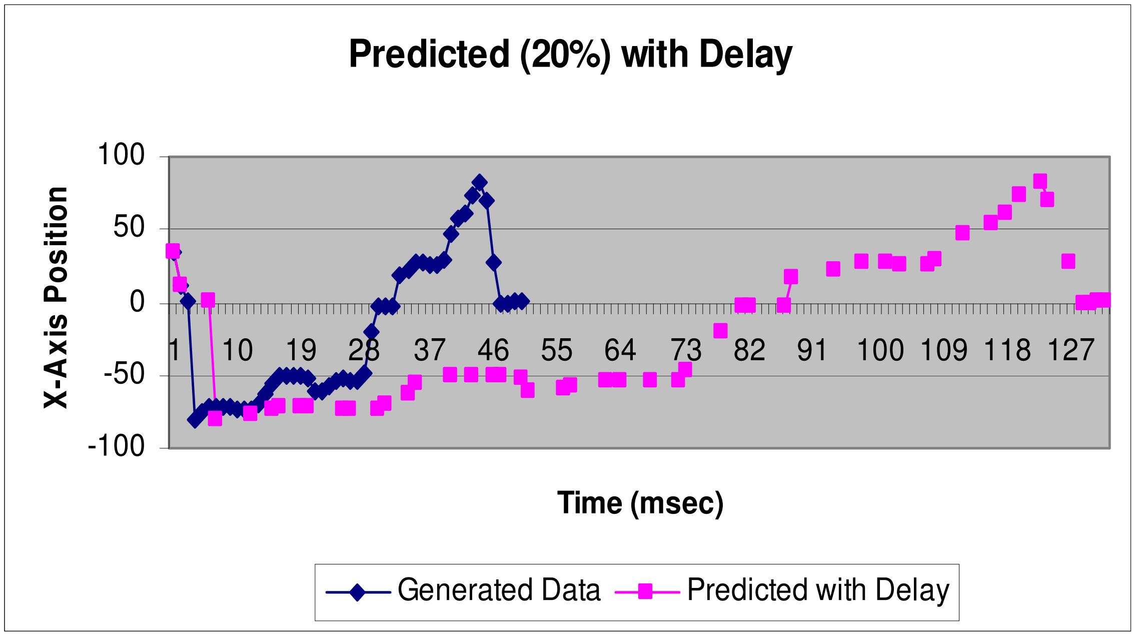


Figure 5.22 Data Comparison after Linear Prediction Algorithm

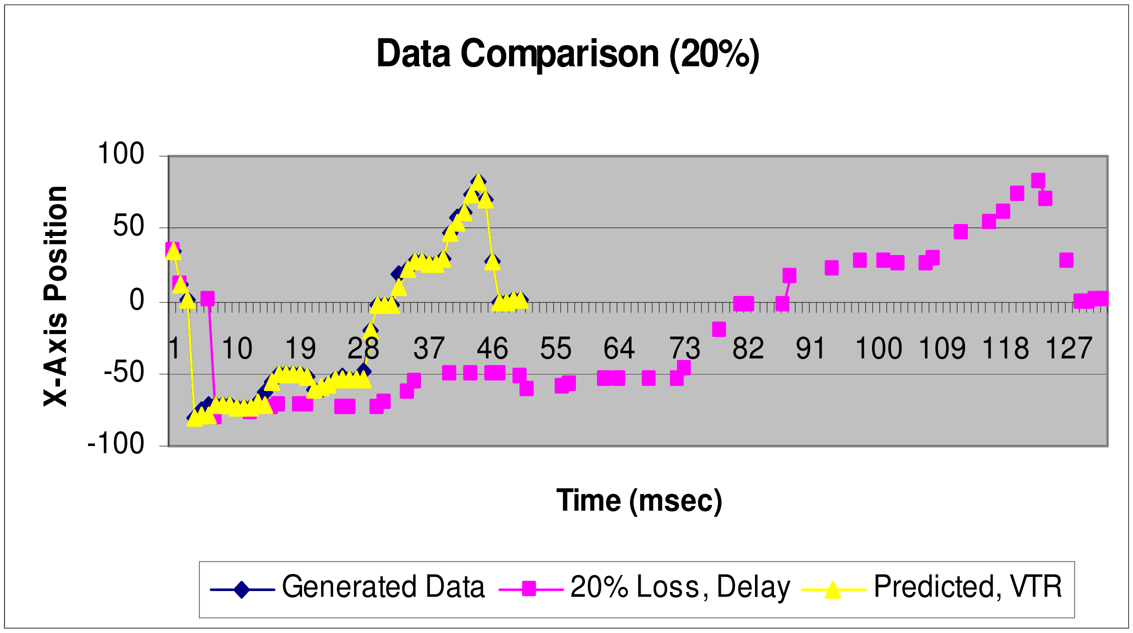


Figure 5.23 Data Comparison after Linear Predictor and VTR

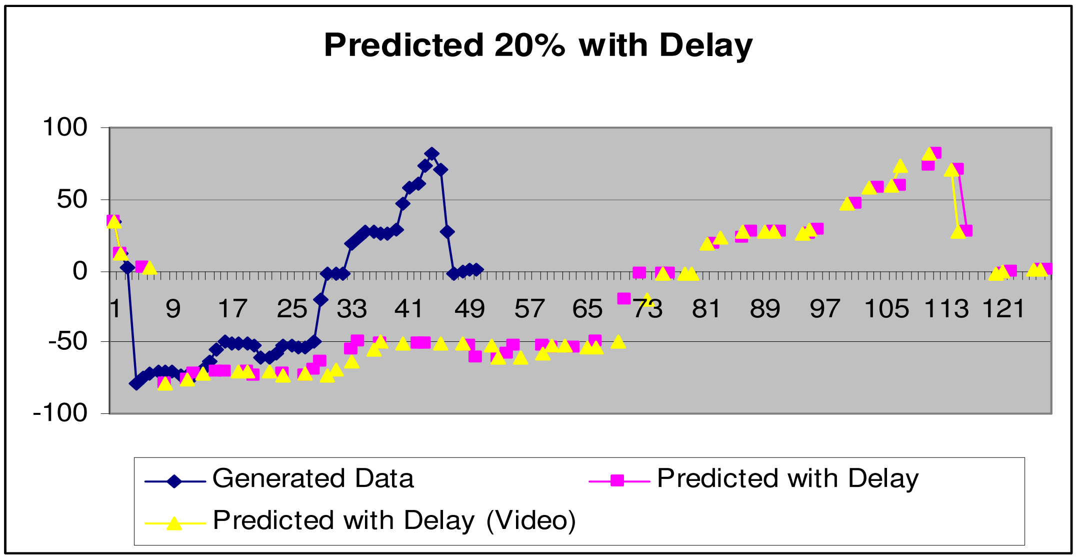


Figure 5.24 Data Comparison after Linear Predictor with Video

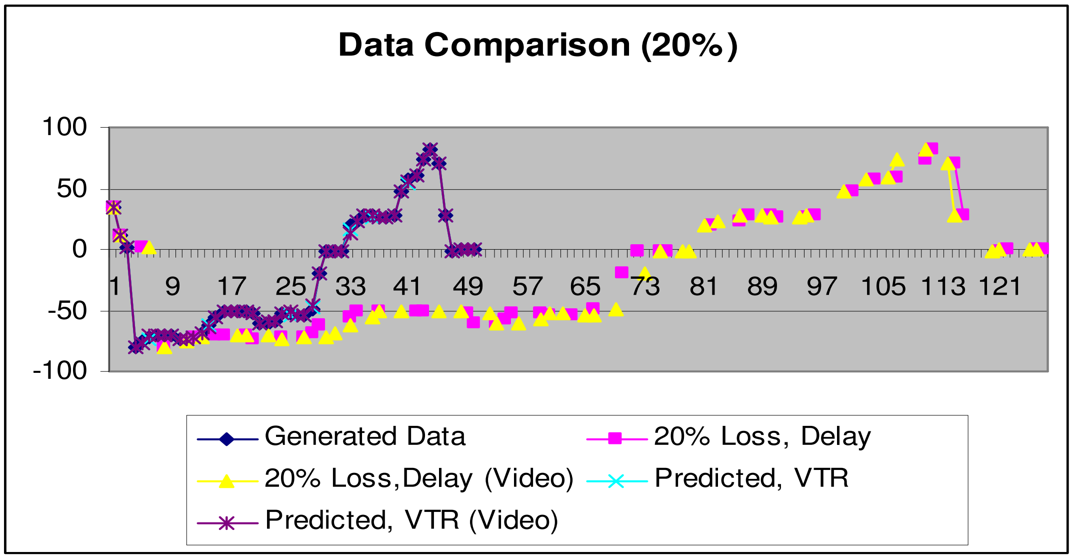


Figure 5.25 Data Comparison after Linear Predictor and VTR (Video)

CHAPTER 6

FUTURE WORK

Force feedback devices and haptic devices are useful in assisting surgeons to perform remote surgery. Our thesis has suggested improving networking techniques that can support such applications. Providing quality of service to the application, the network can improve the intractability. In this section we suggest some future works that can be done to improve the collaborative nature of the application.

Group Synchronization:

Our application dealt with a single source and single receiver performing a 1:1 communication. In real world scenario, we would have multiple requests working on the same object. In such cases, n : 1 communication is required and the network should be able to perform group synchronization. A real world scenario would be group of surgeons operating on the same remote patient. The network should be able to provide synchronized data transmission between the server requests as well as providing intra media synchronization within each media. Another major concern would be to determine the bandwidth impact on the synchronization scheme when introduced with multi media systems.

SURGNET provides efficient synchronization of the force feedback device and shows an emulated surgical application. The future work would be to integrate more media to the architecture and determine the impacts of such on the application.

CHAPTER 7

CONCLUSION

The proposed research gives an integrated compensation technique which synchronizes the rendering update rate at the remote side with efficient algorithms. SURGNET addresses challenges in compensating for the data packet loss as well as the varying delay jitter. The primary goal of the architecture is to address the major issue of providing a complete solution which addresses the various challenges of the network. With an adaptive approach the algorithms efficiently work in reproducing the generated source and SURGNET platform provides architecture to support it.

The architecture was improved by providing intra media synchronization by introducing the feedback video data with the restricted bandwidth and still able to achieve the required synchronization for both force feedback devices as well as the intra media synchronization. Our work was tested for efficiency in a simulated network by considering real time network scenarios and we studied the efficiency of the synchronization algorithms that we have proposed.

BIBLIOGRAPHY

1. Nikolaos Laoutaris and Ioannis Stavrakakis “Intrastream Synchronization for Continuous Media Streams: A Survey of Playout Schedulers” Department of Informatics & Telecommunications, University of Athens, 15784 Athens, Greece
2. Yutaka Ishibashi, Hidehiko Kasugai, and Masaki Fujimoto “An Intra-Stream Synchronization Algorithm for Haptic Media in Networked Virtual Environments” Nagoya Institute of Technology, Nagoya 466-8555, Japan
3. Pietro Arcidiacono, **Paolo Capodieci, *Alessandro Garibbo, ***Claudio Giarrizzo, ºRomolo Grasso, *Annamaria Raviola “DESNET: a SCPC-DAMA network in satellite telemedicine applications” *Marconi Mobile S.p.a. Genova, Via di Francia, 3, 16149, Genova (Italy)
4. Yutaka Ishibashi and Shuji Tasaka “A Synchronization Mechanism for Continuous Media in Multimedia Communications, Department of Electrical and Computer Engineering, Nagoya Institute of Technology, Nagoya **466,** Japan
5. Yutaka Ishibashi and Shuji Tasaka “A Comparative Survey of Synchronization Algorithms for Continuous Media in Network Environments” Nagoya Institute of Technology, Nagoya 466-8555, Japan
6. Itheon Network Emulator (INE) for Windows and INE Compact User Guide V5.0, Itheon Networks Ltd.
7. Masaki Fujimoto and Yutaka Ishibashi “Packetization Interval of Haptic Media in Networked Virtual Environments” Nagoya Institute of Technology, Nagoya 466-8555, Japan

8. Olarn Wongwirat and Shigeyuki Ohara "Haptic Media Synchronization for Remote Surgery through Simulation" Tokai University

9. Jung Kim ,Hyun Kim, Boon K. Tay, Manivannan Muniyandi, Mandayam A. Srinivasan "Transatlantic Touch: A Study of Haptic Collaboration over Long Distance" The Touch Lab Department of Mechanical Engineering and The Research Laboratory of Electronics Massachusetts Institute of Technology Cambridge, MA

10. Monkman. G.J. - An Electrorheological Tactile Display - Presence (Journal of Teleoperators and Virtual Environments) - Vol. 1, issue 2, pp. 219-228, MIT Press, July 1992.

11. Klein. D, D. Rensink, H. Freimuth, G.J. Monkman, S. Egersdörfer, H. Böse & M. Baumann - Modelling the Response of a Tactile Array using an Electrorheological Fluids - Journal of Physics D: Applied Physics, vol 37, no. 5, pp794-803, 2004

12. Klein. D, H. Freimuth, G.J. Monkman, S. Egersdörfer, A. Meier, H. Böse M. Baumann, H. Ermert & O.T. Bruhns - Electrorheological Tactile Elements - Mechatronics - Vol 15, No 7, pp883-897 - Pergamon, September 2005.

13. BioRobotics Laboratory http://robot.kut.ac.kr - Research on Haptics and Teleoperation. Stable Haptic Interaction

14. Jacobus, C. et al., Method and system for providing a tactile virtual reality and manipulator defining an interface device, US Patent 5,389,865

15. Jacobus, C., et al., Method and system for simulating medical procedures including virtual reality and control method and system,US Patent 5,769,640

16. Virtual Haptic Back : http://www.ent.ohiou.edu/~bobw/html/VHB/VHB.html

17. "ActiveX controls—Microsoft papers, presentations, web sites, and books, for ActiveX controls." Available: http://www.microsoft.com/com/tech/ActiveX.asp

18. A. Okamura, C. Richard, and M. Cutkosky, "Feeling is believing: Using a force-feedback joystick to teach dynamic systems," *ASEE J.Engg. Educ.*, vol. 91, no. 3, pp. 345–349, 2002.

19. M. Correia and P. Pinto, "Low-level multimedia synchronization algorithm on broadband networks," in *Proc. ACM Multimedia '95,* pp. 423-434, Nov. 1995.

20. K. Ravindran and V. Bansal, "Delay compensation protocols for synchronization of multimedia data streams,"IE& Trans. Knowl. andData Eng., vol. *5,* no. 4, pp. 574-589, Aug. 1993

21. **Y.** Ishibashi and **S.** Tasaka,"A synchronization mechanism for continuous media in multimedia communications," in Proc. IEEE INFOCOM'95, pp. 1010-1019, Apr. 1995.

22. D. P. Anderson and G. Homsy, "A continuous media I/O server and its synchronization mechanism," IEEE Computer,

23. L. Li, A. Karmouch and N. D. Georganas, "Synchronization in real time multimedia data delivery," in Con$ Rec. IEEE ICC'92, pp. 587-591, June 1992

24. Y. Xie, C. Liu, M. **J.** Lee and T. N. Saadawi, "Adaptive multimedia synchronization in a teleconference system," in *ConJ* Rec. IEEE ICC'96, pp. 1355-1359, June 1996.

25. K. Rothermel and T. Helbig, "An adaptive protocol for synchronizing media streams," ACM/Springer Multimedia ***Sys***tems, vol. *5,* no. *5,* pp. 324-336, Sep. 1997

26. A. La Corte, A. Lombardo, **S.** Palazzo and G. Schembra, "A feedback approach for jitter and skew enforcement in multimedia retrieval services," in *Con$* Rec. IEEE GLOBE- [34] A. Ichikawa, K. Yamaoka, T. Yoshida and Y. Sakai, "Multimedia synchronization system for MPEG video based on quality of pictures," in Proc. IEEE Multimedia Systems '96, COM'9.5, pp. 790-794, NOV. 1995.

27. Y. Ishibashi, E. Minami and **S.** Tasaka, "Performance measurement of a stored media synchronization mechanism: Graceful recovery scheme," IEICE Trans. Commun., vol. E79-B, no. 3, pp. 399-41 **1,** Mar. 1996.

28. Y. Ishibashi. **S.** Tasaka and A. Tsuii. "Measured Derformance pp. 81 1-817, NOV. 1995. of a live media synchronization mechanism in an ATM network," in Con$ Rec. IEEE ICC'96, pp. 1348-1354, June 1996.

29. Y. Ishibashi and **S.** Tasaka, "A media synchronization mechanism for live media and its measured performance," IEICE Trans. Commun., vol. E81-B, no. 10, pp. 1840-1849, Oct. 1998.

30. **S.** Tasaka, **Y.** Ishibashi and H. Imura, "Stored media synchronization in wireless LANs," in Con$ Rec. IEEE GLOBECOM' 96, pp. 1904-1910, Nov. 1996.

31. **Y.** Ishibashi, **S.** Tasaka and T. Okuoka, "A media synchronization mechanism for MPEG video and its measured performance," in Proc. ***13th*** International Conferenceon Computer Communication, pp. 163-1 70, Nov. 1997.

32. **S.** Tasaka, H. Nakanishi and Y. Ishibashi, "Dynamic resolution control and media synchronization of MPEG in wireless LANs," in Conf Rec. IEEE GLOBECOM'Y7, pp. 138-144, Nov. 1997.

33. S. Tasaka and Y. Ishibashi, "Stored media synchronization schemes in ATM and wireless networks: A performance comparison", in Proc. IEEE ICUPC'Y7, pp. 766-772, Oct. 1997.

34. **S.** Tasakaand Y. Ishibashi, "Media synchronizationin heterogeneous networks: Stored media case," IEICE Trans. Com*mun.,* vol. E81-B, no. **8,** pp. 1624-1636, Aug. 1998.

35. **S.** Tasaka and Y. Ishibashi, "A performance comparison of single-stream and multi-stream approaches to live media synchronization,'' IEICE Trans. Commun., vol. E8l-B, no. 11,

36. K. Fadiga, Y. Ishibashi and **S.** Tasaka, "Performance evaluation of a dynamic resolution control for video traffic in mediasynchronized multimedia communications," IEICE Trans. Commun., vol. E81-B, no. 3, pp. 565-574, Mar. 1998.

37. M. Kato, **N.** Usui and **S.** Tasaka, "Performance evaluation of stored media synchronization in PHS," (in Japanese), Trans. IEICE, vol. J80-B-11, no. 9, pp. 749-759, Sep. 1997.

38. M. Kato, N. Usui and **S.** Tasaka, "Performance evaluation **of** live media synchronization in **PHS,"** (in Japanese), Trans. IEICE, vol. 581-B-11, no. **8,** pp. 762-772, Aug. 1998.

39. . Kato, N. Usui and **S.** Tasaka, "Media synchronization control based on buffer occupancy for stored media transmission in PHS," IEICE Trans. Fundamentals, vol. E81-A, no. 7, pp. 1378-1386, July 1998. pp. 1988-1997, NOV. 1998. Also, IEEE ICC'98.

40. R. Ramjee, J. Kurose, D. Towsley and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proc. IEEE INFOCOM'94,* pp. 680- 688, June 1994.

41. **S.** Cen, C. Pu,, R. Staehli, C. Cowan and J. Walpole, "A distributed real-time MPEG video audio player," in *Proc. NOSSDAV'95,* pp. 142-153, Apr. 1995

42. L. Lamont, L. Li, R. Brimont and N. D. Georganas, "Synchronization of multimedia data for a multimedia news-ondemand application,"IEEEJ. Sel. Areas in ***Commun.,*** vol. 14, no. 1, pp. 264-278, Jan. 1996

43. **S.** K. Jha and M. Fry, "Continuous media playback and jitter control," in *Proc. IEEE Multimedia Systems'96,* pp. 245-252, June 1996

44. Y. Xie, C. Liu, M. **J.** Lee and T. N. Saadawi, "Adaptive multimedia synchronization in a teleconference system," in *ConJ* Rec. IEEE ICC'96, pp. 1355-1359, June 1996.