# NOTE TO USERS

This reproduction is the best copy available.

# VIDEO AND AUDIO FUSION FOR STREAMING APPLICATIONS

by

Wei Lu

Bachelor of Science, Computer Science
Ryerson University, 2001

A thesis

presented to Ryerson University

in partial fulfillment of the

requirement for the degree of

Master of Applied Science

In the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2004

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

| Name | Address | Date | Signature |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Video and Audio Fusion for Streaming Applications

Wei Lu
Master of Applied Science
Graduate Department of Electrical and Computer Engineering
Ryerson University
2004

# Abstract

This thesis explores a technique for the fusion of streamed audio and video services for real-time applications. It discusses certain novel techniques used to overcome the problems with video and audio synchronization over the Internet of a tele-bot. We have developed a demonstration called the WAX, at the Network-Centric Applied Research Team (N-CART) laboratory located within the School of Computer Science at Ryerson University. WAX is equipped with an onboard camera and a microphone, as well as a 2.4 GHz wireless transceiver for transmitting video and audio feeds, and at the same time receiving commands from the WAX robot server. By launching a web browser and loading the Java client applets, a user can see as well as hear what is around WAX in near real-time, while being able to move the robot around its environment.

# Acknowledgments

I would like to acknowledge all the invaluable help from my supervisor Dr. Alex Ferworn. Without whom this would not have been possible.

I would like to thank all my committee members: Dr. Alireza Sadeghian, Dr. Marcus Santos, and Dr. Dimitri Androutsos. In addition to their invaluable time spent on my thesis, their insightful comments and suggestions enhanced the professionalism of this research. I would also like to thank Dr. Denise Woit, although she is not in my committee, but she is always willing to help.

I am grateful to my friends and colleagues from Ryerson University, who have remained my friends even through my disappearance into my thesis work.

Last but not least, my thanks go to my family. Without their encouragement, support, understanding, sacrifices and love, none of this would have been possible.

# Table of Contents

# List of Figures

# 1. Introduction

The word *Robot* was first introduced in 1923 when the Czech writer Karel Capek's play, *Rossum's Universal Robots (R.U.R)*, was presented in London and New York [1]. R.U.R was published in 1921, and later was translated into English. Since that time humans have been fascinated with robots. Robotic research has gained great momentum since the seventies when enabling technologies became available. "Robot" was derived from the Czech word, "robota", meaning "servitude, forced labor", and they have been serving humanity from the time they became technically feasible. Although robotic technology is young, it is growing at a rapid rate. Factories were early adopters in the use of robotic machinery because it could handle heavy materials, complex and repetitive maneuvers, and is more efficient for performing certain specific tasks involving highly accurate placement, such as positioning transistors on a circuit board. Robots are also slowly entering the consumer market. Many companies are spending millions of dollars on robotic research, such as the development of the entertainment robot AIBO [2] from SONY and the Humanoid robot ASIMO [3] from HONDA. In the near future, robots could be in every household as computers are today, with capabilities well beyond providing entertainment.

At the moment, robotic technology and telecommunication allow the remote control of robots. This is known as teleoperation, which takes several forms and can be accomplished via any communication medium, wired or wireless. Due to the availability and low cost of the Internet, many applications use Internet-based teleoperation. However, the Internet is an unreliable medium compared to dedicated communication links, such as direct one-to-one

wired connections. This is problematic for teleoperation because the Internet connection is always subject to random delay, loss of information, and even complete loss of connection. When the Internet is considered as the transmission medium, time delay becomes an important factor in the stability, efficiency, and safety of real-time robotic control from a distance.

For real-time teleoperated systems, the operator is able to control the remote device, at the same time receiving feedback from the robot. The feedback provides the operator with a sense of what is happening remotely, allowing the operator to indirectly interact with the remote task. The effectiveness and realism of the feedback is known as the transparency of the system [4]- [7]. In the past decade, to compensate for time delay, research has been conducted in the realm of telepresence to ensure the transparency, stability, efficiency and safety of real-time teleoperation systems by providing visual information from the remote site to the operator [8]-[11]. Telepresence occurs when "the human operator receives sufficient information about the teleoperator and the task environment, displayed in a sufficiently natural way, that the operator feels physically present at the remote site" [9]. However, the research has several limitations that cannot be addressed using visual information alone but requires auditory information as well [11], [12], [15], [16]-[41]. In addition, synchronization of video and audio streams in such systems has not been addressed.

Much of the synchronization work found in the literature relates to multimedia synchronization in which the emphasis is on compression efficiency and quality, but this does not address the safety and time-delay issues of real-time applications. Therefore multimedia synchronization is not suitable for real-time teleoperated systems.

This thesis presents a general framework for the modeling, analysis, and design of video and audio enabled Internet-based real-time teleoperation systems. In addition, issues related to effective synchronization are addressed. Design guidelines to ensure compatibility, stability, and synchronization are detailed, and the advantages of features are illustrated using the developed concepts.

This chapter discusses a broad overview of teleoperation systems and the motivation for video and audio enabled Internet-based real-time multimedia teleoperation. Problems and limitations of current Internet-based teleoperation system are also presented.

## 1.1 Introduction to Teleoperation

Teleoperation is the remote control of devices, which typically are robot manipulators [9]-[10]. Teleoperation has many benefits, not the least of which are improved safety and security for humans. It has been applied to many applications including:

- **Search and rescue**: Mobile robots can be used to search areas which are inaccessible and/or dangerous for humans. Various mobile robots were used during the search after the World Trade Centre (WTC) catastrophe [12] [13].
- **Security and law enforcement**: Many buildings are equipped with remotely monitored and controlled cameras to monitor areas of the building from a central location.
- **Hazardous material manipulation and removal**: Teleoperated robots are well suited to this kind of task, reducing the need for humans to actually visit an area. Radioactive, chemical, or explosive substances are just too dangerous for humans to manipulate safely. For example, teleoperated robots are used for chemical spill clean up or bomb detonation [14].

- **Remote exploration**: Since robots don't have the limitations that are imposed on humans, such as sensitivity to pressure, heat, and light, they are more flexible about working environments, and provide safety for humans. NASA's Mars Pathfinder Mission is a good example of this type of application [15].
- **Surveillance**: Surveillance using teleoperation is not a new concept, and it has been extensively developed. The U.S. has engineered many Unmanned Aerial Vehicles (UAVs) including the RQ-1 Predator [22] which is used in a remote reconnaissance role.
- **Traffic control**: With today's technology, trains and subways use teleoperated routing systems [72].

## 1.2 Current Approach

Despite all the applications, the human experience during teleoperation is still far from that of "being there" due to limited remote sensory capabilities. Current research mainly focuses on the effectiveness of video-only teleoperation [8] – [11].

To provide transparency to the system, telepresence is used to reassemble the remote environment in an understandable representation of reality for the operator, the senses such as visual, auditory, temperature, or many other can suffice. For simple teleoperation tasks, such as train routing, telepresence is not needed, because the only information the operator needs is whether the operation was successful or not.

For any complex task such as navigation, telepresence becomes essential. Telepresence not only makes such teleoperation tasks possible, it also helps to ensure the accuracy, efficiency, and safety of the operation.

Teleoperation, in general, can be accomplished via any kind of communication media. However, since the Internet has matured into what we know today, much research has been focused on Internet-based teleoperation [11]-[21]. The use of the Internet has many advantages in comparison to a dedicated communication channel. The Internet is less expensive than using dedicated hardware and technology for the communication. It is also readily accessible and provides wide availability.

Since HTTP (Hypertext Transfer Protocol) is such a common protocol employed on the Internet, many teleoperated robots use HTTP and hence World Wide Web (WWW) for their interface and control. Although Internet-based robots are not necessarily web-based, the web provides excellent support for multimedia and user interaction which provide attractive research in Internet-based robots and teleoperation in general.

Unfortunately, the use of the Internet for teleoperation also has some drawbacks. The Internet, in its current state, is not designed and implemented to handle real-time applications such as teleoperation. The Internet does not offer any guarantees for timely delivery of feedback required for real-time operations, and the delay is unpredictable [28]-[32]. Therefore, it is a challenge to maintain stability, transparency, and safety of teleoperation over the Internet.

In addition, human experience duri' Internet-based teleoperation is still far from the real thing, often offering only video feedback. Other sensory information, particularly auditory information is missing from many Internet-based teleoperation systems. Audio can provide useful information to operators, such as human speech, the environment and audible events at the remote site.

Audio is also important in many critical situations such as human rescue, bomb disposal, and other operations. If there is a survivor who is trapped somewhere inside a collapsed building, a rescue robot can be sent out to search in areas that are either too small or too difficult for human reach [12] [13]. If the robot can capture and transmit audio, the survivor can call out for help. Audio feedback may also be used to guide the operator in navigating the robot through the hostile environment.

However, there are several challenges for enabling audio in Internet-based real-time teleoperated systems. Many Internet-based teleoperated systems use Java as their interface. Java is the platform-independent Virtual Machine (VM) which allows server-client communication on the web. Hence it becomes the choice for implementation of teleoperation clients. However, Java has limited support for audio and streaming in general. In addition, as a general communication medium, the Internet does not provide any guarantee for the timely delivery of streams needed for real-time teleoperation. Therefore the synchronization for both video and audio streams over the Internet is difficult to achieve.

## 1.3 Contributions

The goal of this thesis is to develop an Internet teleoperated system which provides both visual and auditory information for teleoperation. This thesis attempts to:

- Develop a video and audio synchronization algorithm for Internet-based real-time teleoperation system. Adding audio to existing video-only teleoperation systems requires synchronization. There are many techniques for synchronizing video and audio; however, they are designed for different purposes. For example, some perform better with stored media while others are made for real-time streaming.

- Attempt to overcome the limitation of audio streaming in Java. The Java Application Programming Interface (API) does not have any support for audio streaming applications. The only audio class it provides is to play a stored audio file from the server. The thesis explores the possibility of allowing audio streaming on Java-enabled web browsers.

WAX is the implementation of the proposed Internet teleoperated system. It is based on the existing teleoperated robot MAX [42] - [44]. WAX can provide both video and audio feedback to the operator, using a Java applet interface. Both streams are used to help the operator to interpret the remote environment and determine the robot's status and context, and therefore help the operator to make an informed decision in controlling the robot. To ensure successful teleoperation even under low-bandwidth connections, the WAX system allows the operator to enable and disable the audio at any time, as well as changing the video resolution.

## 1.4 Thesis Organization

The thesis is organized as follows: Chapter 2 is the literature review. Current streaming technologies are examined. Several possible approaches are examined and compared. Chapter 3 introduces the algorithm and system design for a novel video and audio enabled teleoperated system called WAX. It also introduces a new video and audio synchronization mechanism for real-time communication. Chapter 4 provides the detailed implementation of WAX and the working model, as well as the experiments conducted. Chapter 5 gives conclusions and suggests possible future work.

# 2. Literature Review

## 2.1 Overview of Multimedia Streaming

The video of Internet-based robots is sent to the remote operator through a process called video streaming. The video is processed on a computer connected to the robot, wired or wirelessly. The computer can also be on-board. This computer acts as a video streaming server, which streams live video data to the operator's computer upon request. The operator's control program, in turn acts as a client, which queries the server for current video. If audio is added to the current video-only streaming model, the process becomes multimedia streaming, since more than one medium is used.

Two types of multimedia streaming exist and are in common use today: On-demand streaming and real-time streaming. On-demand streaming allows a client to download a static file stored on a server. Most video players employ this strategy and allow the client to start playing the video without waiting for the entire file to be downloaded.

Some on-demand streaming providers claim to be real-time, but in fact they are merely referring to the continuity of the stream being real-time. In other words, their bandwidth is sufficient for the video to be played without interruptions once a sufficient amount of the file has been received. Movie trailers from apple.com are a good example of an on-demand streaming application.

Real-time streaming usually deals with the broadcast of live action. Some real-time streaming applications include TV and radio station broadcasts. Many teleoperation applications employ real-time streaming as well. Real-time streaming tends to be quite

sensitive to any delay in transmission as the technique does not allow for the luxury of buffering. Websites such as real.com provide many live radio stations.

## 2.2 Multimedia Synchronization

Synchronization is crucial to a multimedia system. It maintains the temporal relationship between streams, and provides a more meaningful and enjoyable multimedia presentation for human viewers. There are two approaches to synchronize multiple streams: single-stream and multi-stream.

The single-stream approach involves multiplexing individual media streams into one transport stream at the source location [73]. In the single-stream approach, temporal alignment between media streams are predefined and embedded in the transport stream. This approach is used within many popular media standards, especially in the entertainment market, such as ISO/IEC 11172-1 [74] for MPEG-1, ISO/IEC 13818-1 [75] for MPEG-2 and ITU-T H.233 [76].

The multi-stream approach sends each media stream individually. This approach is typically used for real-time streaming applications, including teleoperation. The Multi-stream approach has been adopted by many streaming standards, such as RFC 1889 for Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol [77], as well as MBone video and audio broadcasting services [78], [79]. The multi-stream approach is faster in delivery of the streams compared to the single-stream approach, since no multiplexing time is required. It is also more flexible than the single-stream, the number of media streams can be increased or reduced to compensate for network delay and user needs. This thesis will employ the multi-stream approach.

Multi-stream synchronization can be divided into three basic steps:

- Master and slave streams selection,
- intra-stream synchronization, and
- inter-stream synchronization.

## 2.3 Master and Slave Streams

To synchronize multiple individual streams, one master stream must be chosen as the reference [62]. Other slave streams can then be synchronized with the master stream. It is also possible for a slave stream to become a master stream for other slave streams. Generally, the stream with the most sensitivity to synchronization error and a higher sampling rate is used as the master stream. Research has shown that a smooth unbroken audio stream is required for speech transmission and ease of recognition by humans, while humans are quite tolerant of "jittery" video [6], [71]. With this in mind, we have chosen the audio stream to be the master stream and video to be the slave stream.

## 2.4 Intra-Stream Synchronization

Intra-stream synchronization is the synchronization of the stream itself. An intra-stream asynchrony can occur when the network delay is longer than the size of a single stream segment, also known as the media unit (MU). For example, if each audio MU is about one second, then the network delay must be less than one second, otherwise there would be a gap, or time delay between previous and current audio clips. Therefore the stream segment size must be long enough to compensate for the end-to-end time delay on the network--in our case the Internet.

For images, intra-stream synchronization is not an issue, since the images are discrete snapshots and not continuous. On the other hand, audio streams are continuous. Hence an appropriate stream segment size has to be chosen to ensure the smooth audio playback at the client side over the Internet without too much delay.

## 2.5 Inter-Stream Synchronization

Inter-stream synchronization is the synchronization of slave streams to the master stream. Therefore, inter-stream synchronization is not required for the master stream. Inter-stream synchronization is carried out after intra-stream synchronization is complete.

Sound clips are recorded in a different way than video images. For video images, each query to the video hardware returns the entire snapshot of the image. On the other hand, audio is in a continuous waveform which must be digitized by the sound hardware before it can be processed by the computer. Digitizing the audio is the process of assigning numerical values to sample points in the audio waveform. The numerical value can be an 8-bit or 16-bit integer. The density of the sampling points is known as the sampling rate of the audio which is set by the sound hardware. Higher sampling rates give more detail of the audio waveform but require more bandwidth as well. Each audio channel has its own audio waveform. A mono sound has one audio channel, while stereo sound has two. The bitrate of the audio is determined by the product of sampling value, sampling rate and audio channel. For example, the bitrate for 8-bit, 8 Kilohertz (KHz, sampling rate) mono sound is 64 kilobits per second.

Since video images are discrete and sound clips are a continuous "wave", the synchronization between them requires special attention. In addition, inter-stream synchronization is more challenging for an adaptive teleoperated system, where stream

segments can be skipped at anytime to compensate for network delay. However segments from any stream can also be missing at anytime, making inter-stream synchronization difficult.

## 2.6 Virtual Clocks

Virtual clocks are often referenced in media synchronization. A virtual clock is used when playing back a media stream. The beginning of the stream is the start of the virtual clock. In other words, a virtual clock acts as a stopwatch for media playback, and it is used as a temporal reference for intra-stream and inter-stream synchronizations.

## 2.7 Existing Real-Time Synchronization Techniques

Due to the unreliability and lack of support for guaranteed delivery on the Internet, the task of synchronization control must be moved to the application level. A variety of application-level algorithms have been proposed to achieve the intra-stream and inter-stream synchronization [62]-[113]. These algorithms differ in terms of goals and application scenarios. In general, they can be categorized into four types of synchronization control techniques,

- basic control,
- preventive control,
- reactive control, and
- common control.

These techniques can be used alone or in combination with others to achieve the desired synchronization for the targeted applications. Note the terms "technique", "method"

and "algorithm" used in this thesis. A method is used to carry out a particular technique, or approach for synchronization. An algorithm can employ one or more techniques for a particular application. Hence, a method can also be one algorithm.

## 2.8 Basic Control

Basic Control is the simplest control of synchronization techniques, and it is often used in conjunction with other techniques to achieve optimal results. Basic control adds synchronization information, such as timestamping, sequence numbering, etc to each segment of the stream [62].

### 2.8.1 Attachment of Synchronization Information

At the source location, timestamps, sequence numbers or any other synchronization markers [80] are generated and attached to the MUs. The timestamp is usually the generation time of the MU. When the generation of MUs is periodic and non-skipping, timestamps can be replaced by sequence numbers [81]. In addition, it is easier to use sequence numbers to manage inter-stream synchronization by comparing the sequence numbers of each stream.

### 2.8.2 Buffering

To organize the MUs in the proper order, the destination temporarily stores MUs in the destination buffer to compensate for network delay. It then outputs them according to the synchronization information [80].

If Transfer Control Protocol/Internet Protocol (TCP/IP) is used as the mechanism for transferring data, then large buffering is not required, since TCP guarantees the packets will arrive at their destination in the proper order and manages retransmission if packets are lost.

## 2.9 Preventive Control

The second technique is preventive control, which uses algorithms to avoid asynchrony, such as adjusting the destination buffering time of the MUs based upon an estimation of network delay. This technique can be used by media players to prevent pauses during playback.

### 2.9.1 Transmission of MUs According to Synchronization Information

For stored media, MUs are transmitted based on their synchronization information such as timestamps [82]. For example, video frames are sent according to their frame intervals. This technique is not applicable for live media because the timestamp is the generation time of the MU, and the MU is ready to be transmitted as soon as it is captured. This technique is not effective on high-latency networks, because MUs have to be sent ahead of time to compensate for network delay.

### 2.9.2 Priority-Based Transmission

The source schedules the transmission of stream MUs based on their deadline requirements [83], [84]. The deadline requirements can be the stream's MU size, output deadline, and sensitivity to asynchrony.

### 2.9.3 Source Preventive Skippi · ·ıd Pausing

The source skips MUs which do not meet their deadline for sending, or selectively choose a MU from the queue with the appropriate timestamp [42] – [52]. This method is very effective in sending up-to-date MUs even under network bandwidth constraints. However, it is very difficult to achieve intra-stream synchronization and inter-stream synchronization with this method because MUs can be missing from any stream at any time.

### 2.9.4 Interleaving of MUs

MUs from streams are interleaved and combined into one transport stream [74] – [76]. This is the same scheme used for single-stream synchronization and many stored media formats, such as MPEG. This technique improves the quality of inter-stream synchronization. However, it may degrade intra-stream synchronization quality of the media streams that are sensitive to delayed or jittery transmission unless appropriate output buffering is used.

### 2.9.5 Destination Preventive Skipping and Pausing

The destination discards MUs or pauses (repeats) output of MUs depending on the length of the output queue [85] - [88]. It is possible to insert dummy (noise) data instead of pausing. When buffer occupancy at the destination is faster than the playback rate, MUs can be discarded at an interval to match the playback rate, such as skipping one MU for every few MUs [87]. This algorithm is not as efficient as source preventive skipping and pausing, since bandwidth is already wasted for discarded MUs.

### 2.9.6 Buffering Time Adjustment with Network Delay Estimation

The destination changes the buffering time of MUs according to estimated or measured network delays [89] - [92]. Intra-stream synchronization is achieved by choosing the appropriate buffering time which is larger or equal to the maximum network delay. This is a common and simple preventive control algorithm.

## 2.10 Reactive Control

The third category is reactive control, which maintains synchronization based on detection or prediction of asynchrony. The approaches include the use of reactive skipping and pausing, shortening or extending playback duration, and playback delay control.

### 2.10.1 Transmission Rate Adjustment

The transmission rate of MUs can be adjusted according to the amount of asynchrony among media streams [93]. When the destination detects asynchrony (delay) between streams, it sends feedback information to the source. The source then adjusts the transmission rate to match the reception rate to prevent asynchrony, i.e., by reducing video frames per seconds.

### 2.10.2 Reduce Number of Media streams Transmitted

The source reduces the number of media streams transmitted when it is difficult for the destination to recover from asynchrony [94], [81]. For example, the source can stop the transmission of a video stream temporarily to allow the destination to recover from asynchrony, and then the source can restart the video transmission.

### 2.10.3 Shortening and Extending of Output Duration

In order to recover from asynchrony gradually without noticeable degradation, the destination shortens or extends the output duration of MUs until synchronization is recovered [81], [82], [89]. For example with video stream, if the MU for the next frame is late, the frame rate decreases to compensate for the delay. If a burst of the previous frames is received due to network jitter, then the frame rate is temporarily increased. Network jitter is the fluctuation in network transfer speed, which occurs when network congestion causes some network packets to be held temporarily in the network queue. This often produces a pause in transmission and then a sudden increase in transfer speed.

### 2.10.4 Reactive Skipping and Pausing

When the destination detects asynchrony, it discards or pauses (repeats) MUs until synchronization is recovered [66], [77], [94], [103], [109]. It is different from preventive skipping and pausing, which prevents asynchrony before it occurs, while reactive skipping and pausing recovers synchronization when asynchrony is detected. Preventive control is often used for destination buffer management, while reactive control provides synchronization recovery. Therefore it is possible to deploy both preventive and reactive skipping and pausing for synchronization control.

### 2.10.5 Playback Delay Expansion and Contraction

The destination adjusts the playback delay to match the network delay [81]. [82]. For example, if the destination playback delay time is 100 ms but the current network delay is 200 ms, then the destination increases playback delay time to 200 ms or more. In other words,

the destination waits for 200 ms before outputting MUs to ensure it can playback the stream without interruption. If network delay is low, the playback delay can be shortened.

### 2.10.6 Master-Slave Switching

When the amount of asynchrony for a slave stream becomes large, then the destination switches the stream to master and performs the appropriate adaptation [66], [89].

## 2.11 Common Control

The last category is common control, which can be used to prevent asynchrony as well as provide synchronization recovery. Most common control methods adjust parameters in the system to dynamically adapt to network bandwidth limitation, and hence enable the system to maintain better synchronization. Adaptive control algorithms include dynamic resolution control of video, and adjustment of input rate [103], [104].

### 2.11.1 Adaptive Source Skipping and Pausing

The source skips MUs or pauses output of MUs according to feedback information from the destination [95]-[97].

### 2.11.2 Advance Transmission with Network Delay Estimation

Either the source or the destination estimates the network delay, and then the source advances the transmission timing of MUs according to the estimates [98] – [102].

### 2.11.3 Input Rate Adjustment

The source adjusts the generation frequency of hardware input devices, such as video cameras and microphones, according to synchronization quality [103]. For images, the source can increase or decrease the frames per second the camera can capture. For audio, the source can be the sampling rate and bit rate.

### 2.11.4 Output Rate Adjustment

The destination adjusts the clock frequency of hardware output devices to match transmission speed, and to achieve inter-stream synchronization [103]. This usually works in conjunction with input rate adjustment at the source. However, the destination can also achieve this alone by adjusting the output rate depending on the length of the output queue [109] – [111].

### 2.11.5 Interpolation of Input Data

Data interpolation can be used to dynamically adjust the effective input rate of data [103]. For example, a frame can be interpolated by comparing previous and next frames.

### 2.11.6 Interpolation of Output Data

Higher output rates can be achieved by interpolation of low frequency data input rates [103]. The use of data interpolation at the destination can reduce the amount of data transmission, while data interpolation at the source can reduce capture overhead and overcome hardware limitations, such as the maximum frames per second (fps) the camera can capture.

### 2.11.7 Dynamic Media Scaling

Dynamic Media scaling [104] such as dynamic resolution control and dynamic quality control of video can be performed according to network loads [105] – [108]. For JPEG images, the width and height of the video frame can be reduced to decrease the bandwidth requirement, or increased to provide better clarity. Also the images can be compressed with different percentage of quality to best match the network bandwidth.

To eliminate sudden resolution changes on the client display, dynamic media scaling such as dynamic resolution control of video can also be achieved at the destination, especially to maintain the final image resolution when the source is also performing dynamic media scaling [104] – [108].

# 3. WAX

## 3.1 What is WAX

WAX is an Internet-based multimedia teleoperated system which allows an operator to remotely control a robot or similar mobile device using a web browser. Video and audio signals are sent from the remote robot to the operator's computer to assist the operator in a safe and effective manner in controlling the robot.

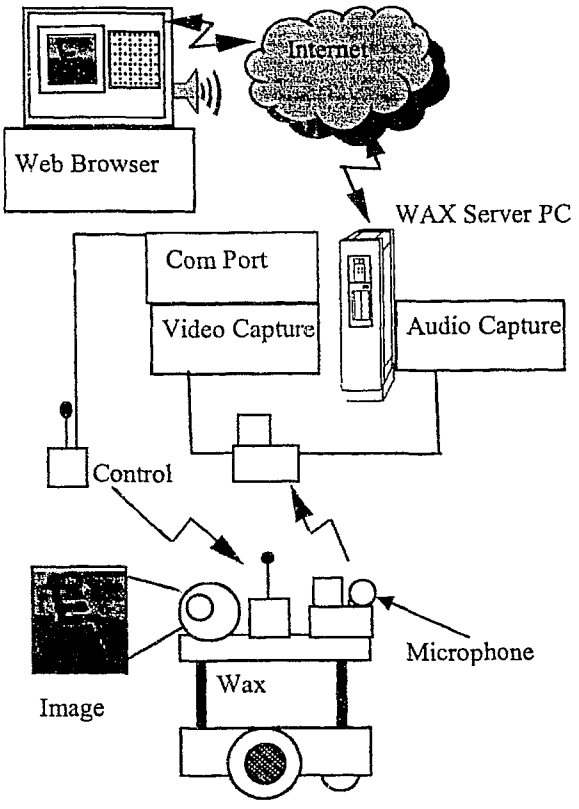


Figure 3.1 WAX System Architecture

WAX consists of two main components: the WAX robot and the WAX server machine. The robot is the physical entity which the operator can control. The WAX server machine is a computer which connects the operator and the robot over the Internet.

The robot is equipped with a camera and a microphone; although, it is possible to use a video camcorder as a video source, it is normally impractical due to their excessive power drain on the limited means of a mobile device.

For teleoperation systems, reliability, durability and timely delivery are normally more important then video quality. Therefore Charge-Coupled Device (CCD) cameras are often used for teleoperated robots. CCD is a camera technology which can produce very good image quality. CCD cameras are commonly known as "webcams." They are also small and inexpensive compared to video camcorders. Typical CCD cameras can achieve a maximum frame rate of about 10 frames per second. This is normally considered sufficient for Internet-based teleoperation since the main bottleneck of the system is network bandwidth, a reduced frame rate is often acceptable and desirable.

Video from the WAX robot's camera and audio from its microphone are transmitted to the WAX server. At the same time the WAX server also sends the control signals from the operator to the robot. This two-way communication can be accomplished through a wired or wirelessly network. Ideally, a wireless link should be used so the robot can move without the encumberance of a tether.

WAX can be powered in a number of ways, such as an on-board rechargeable battery pack, and/or redundantly wired to an off-board power supply.

The WAX server is responsible for processing the video and audio data into presentable formats for web-based applications. Normally this involves conversion into

JPEG images and PCM (a format used in WAV, AU and other audio file types) audio. The server performs two functions, it listens for incoming client connections and also provides a web page for the operator to download the client program from. The WAX client program is a Java applet that can be used on most computer platforms. Once the operator downloads and runs the client program, the client program makes a connection to the WAX server and retrieves WAX video and audio. It renders the video and plays the audio on the local computer, and sends the operator's control commands back to the WAX server.

The WAX server is able to handle multiple client connections. However, in normal operation, only one operator is allowed to control the robot at a time, other clients are in the control queue and only have the ability to observe the robot with the video and audio transmission. The operator is given a time frame to control the robot before the next client in the queue takes over. The previous operator is then sent to the end of the queue.

## 3.2 MAX

WAX's system architecture is based on an older robot called MAX. MAX is an Internet-based teleoperated system with video support only.

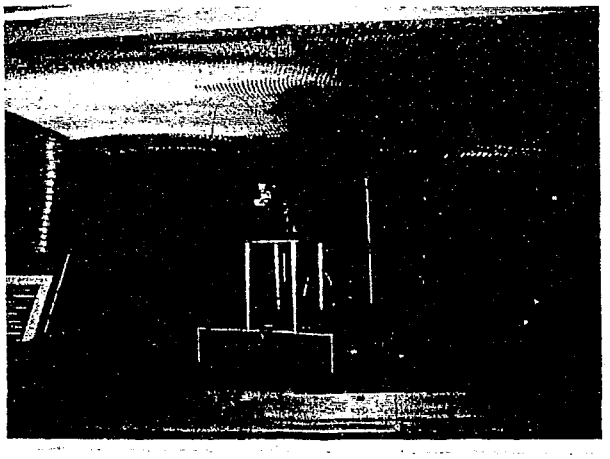

Figure 3.2 MAX

Like WAX, MAX the robot has an on-board CCD camera and receives power in a bumper-car-like manner, employing a number of conducting brushes moving along conducting surfaces above or below MAX. Video signals are wireless and transmitted to a Pentium-class Linux server. Robot control signals are also transmitted wireless from the MAX server to the robot's on-board microcontroller.

Users can control MAX anywhere via a java applet obtained from MAX's website. The interface to MAX is provided by two Java applet windows. One provides a simple control interface and the other provides a stream of JPEGs showing the viewer what MAX is "looking at."

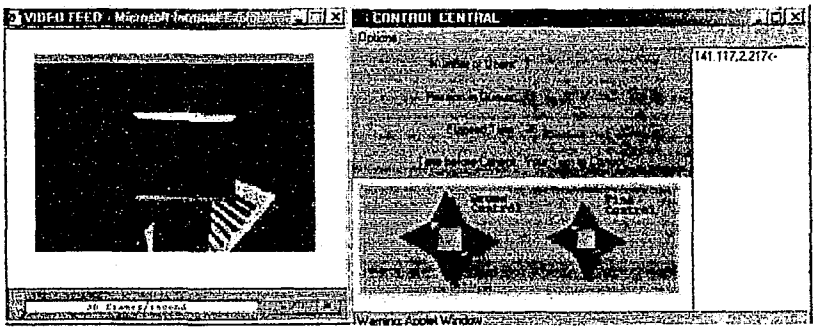


Figure 3.3 MAX Control Interface

The right side of the Control Central window lists all the clients that are currently connected to the robot server. The client on the top can control the robot for 60 seconds and then the control is passed over to the next client in queue.

MAX is able to update the video image almost instantaneously, even under fluctuating network conditions. It achieves this by using source preventive skipping (section 2.9.3) of MUs on its video server.

The MAX video server is a software program that runs under the MAX Linux server machine. The capturing of images is done separately from the MAX video server using

another program. This allows the MAX video server to adapt to different video cards without modification of its code. The capture program captures the video images as soon as the camera can take and store the processed JPEG image to a ram disk for the MAX video server to retrieve.

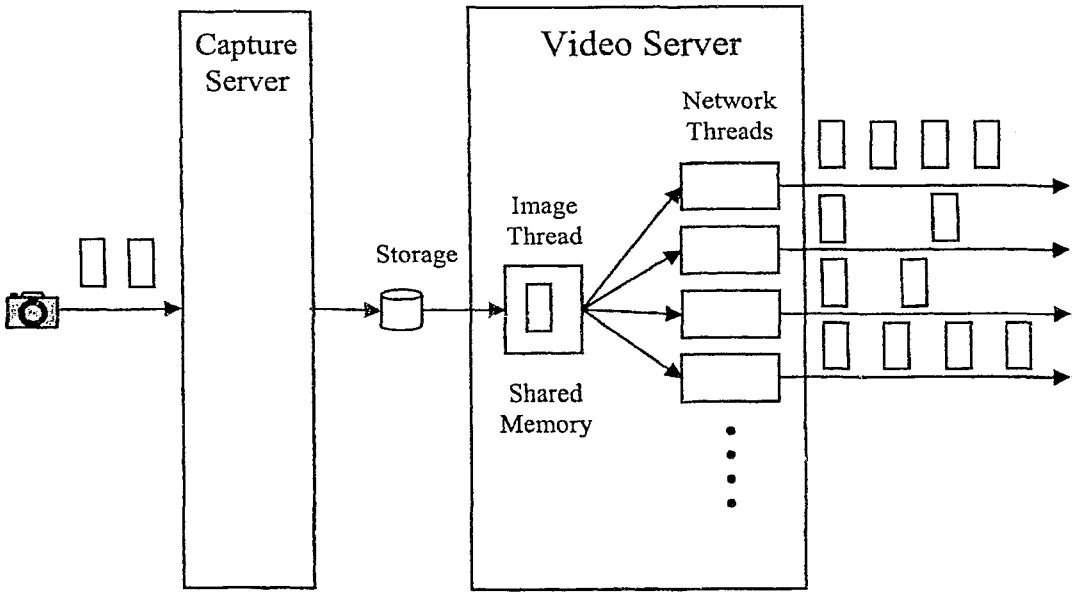


Figure 3.4 MAX Video Server Architecture

The MAX video server uses shared memory under UNIX to store the latest images for its network threads to retrieve. Each network thread handles one operator connection. The network thread only sends the image when the client requests more frames. This allows the network thread to send images at its own speed according to the network conditions at the time of the request. In other words, the video server is able to send streams at different speeds asynchronously to adapt to each client's network bandwidth. Source preventive skipping also guarantees the stream does not overflow the client connection. Therefore., despite each client's current network connection, the operator can always receive the latest images from the robot. Better network connection and bandwidth produce faster image

refresh rates at the client side, up to the camera's hardware limit or predefined maximum refresh rate.

However, the gain in speed and adaptation of preventive skipping compromises the continuity of the video stream. Since MUs from the stream can be skipped at any time to compensate for network delay, the MUs can also be missing at any time. This is not a problem if only a video stream is used. However, when multiple streams are used, source preventive skipping makes inter-stream synchronization difficult. At the same time, timely delivery of MUs is crucial to the effectiveness and safety of teleoperation.

## 3.3 WAX Synchronization Algorithm

WAX attempts to provide a solution by selecting a number of methods from different synchronization techniques. Not all methods are suitable for WAX, especially ones that are designed for stored media. WAX is a real-time teleoperated system, where live sensory information is sent to the operator during mobile operation. Also the methods have to be compatible with the adaptive characteristic of source preventive skipping of WAX.

For reliability and safety of teleoperation, WAX's algorithm ensures timely-delivery first, and then synchronization second. Although, the algorithm is designed for real-time applications, there are limits. For example, it is still inappropriate for high-precision or dangerous operations, such as remote surgery.

Clients might request only the video stream or both streams depending on their need. Therefore, WAX needs to handle each scenario appropriately:

- When only a video stream is requested, it is sent as soon as possible.
- When only an audio stream is requested. The audio stream is sent as soon as possible.

- When both streams are requested, they are sent as soon as possible. However, the video stream needs to roll-back to synchronize with the audio stream. This can create playback delay for the client, which can be a problem when performing actions remotely that require fine coordination without delay.
- When both streams are requested and the bandwidth is insufficient, they are sent as soon as possible, regardless of synchronization. It is a fail-safe mechanism to ensure reliability and safety of the teleoperation as this is a practical means for reducing latency as much as possible given unfavorable network conditions.

When only a video stream is requested, WAX acts just like MAX, which sends the video stream as soon as possible. It is possible to request only the audio stream. However, audio alone is not effective for teleoperation especially for tasks involving navigation or object manipulation; audio may be useful when simply listening to what is happening at the robot's location.

To achieve a continuous audio stream for speech recognition, WAX approaches the problem by first prioritizing the master stream, in this case it is the audio stream. WAX makes an effort to refrain the audio stream from skipping. The video stream is then synchronized with the audio stream using the remaining bandwidth. The audio MU is given a wait time in the source buffer before it is discarded as old material. This wait time is equal to the audio MU generation time. In other words, the audio MU in the buffer is valid until the next audio MU is ready to be sent. If the current MU has not been sent yet, it will be replaced by the audio MU just captured. This is also source preventive skipping. In this way the audio MU in the queue is always the latest possible one. If the client does not have enough bandwidth to receive the MU in time, pauses will occur, that is what WAX algorithm tries to

avoid.

Similar to the video server, the WAX audio server is a program which can handle multiple network clients. Audio capturing is integrated into the audio server, because the sound card interface is much simpler then the video capture card interface, and the Linux Operating System (OS) Kernel already has the API for programs to interact with the microphone. The WAX audio server spawns a new network thread for each connecting client and the network thread is also able to stream the audio at its own pace, independent from other network threads.

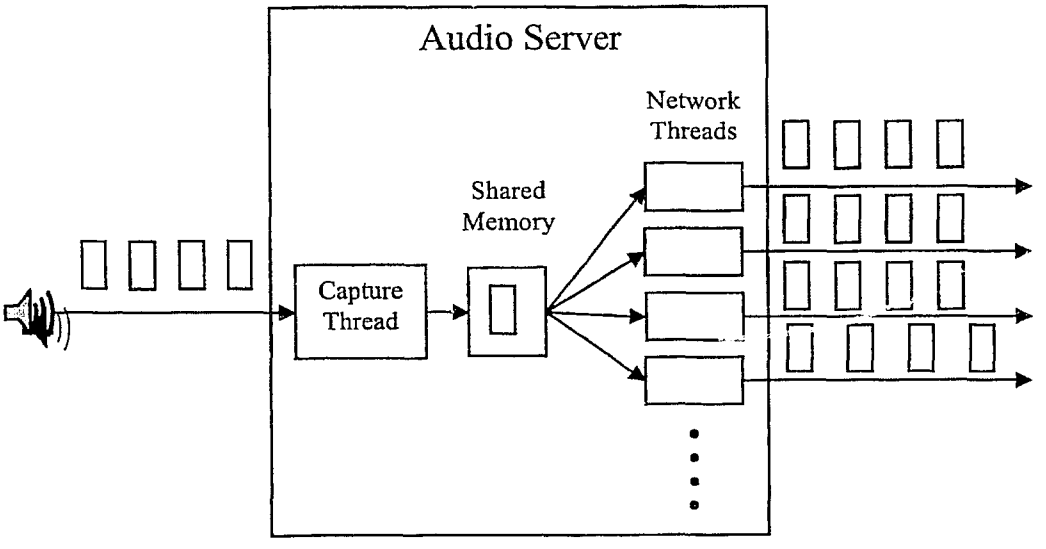


Figure 3.5 WAX Audio Server Architecture

WAX performs intra-stream and inter-stream synchronizations on both the source (server side) and the destination (client side). However, the majority of the work is done at the source, because the server has more information about streams and has direct control of the streaming process. For example, if asynchrony occurs due to excessive MUs, it is more efficient for the server to send less data then let the client discard excessive MUs, that

consume valuable bandwidth.

### 3.3.1 Source Synchronization Control

The WAX server machine runs three software servers in total: the video server, audio server and control server. The video server and audio server are responsible for sending video and audio streams to the client, while the control server is responsible for receiving commands from the client. Since the control stream is going in the opposite direction to the video and audio streams, it is independent of these two streams, and it does not need to be synchronized with video and audio. However, under some circumstances, such as telesurgery [16] – [19], where a control loop requires extremely close synchronization WAX's synchronization technique would be inapplicable and extremely dangerous.

For each client connection, WAX spawns three threads: a video thread, audio thread and control thread. These threads are independent of each other since they are created from different programs. However, the video thread and the audio thread for the same client work closely together to form a logical network unit. The control thread handles the control signals which travel in the opposite direction of video and audio streams. Since the control commands are always executed immediately upon receiving, the control signal does not need to be synchronized with the video and audio streams.

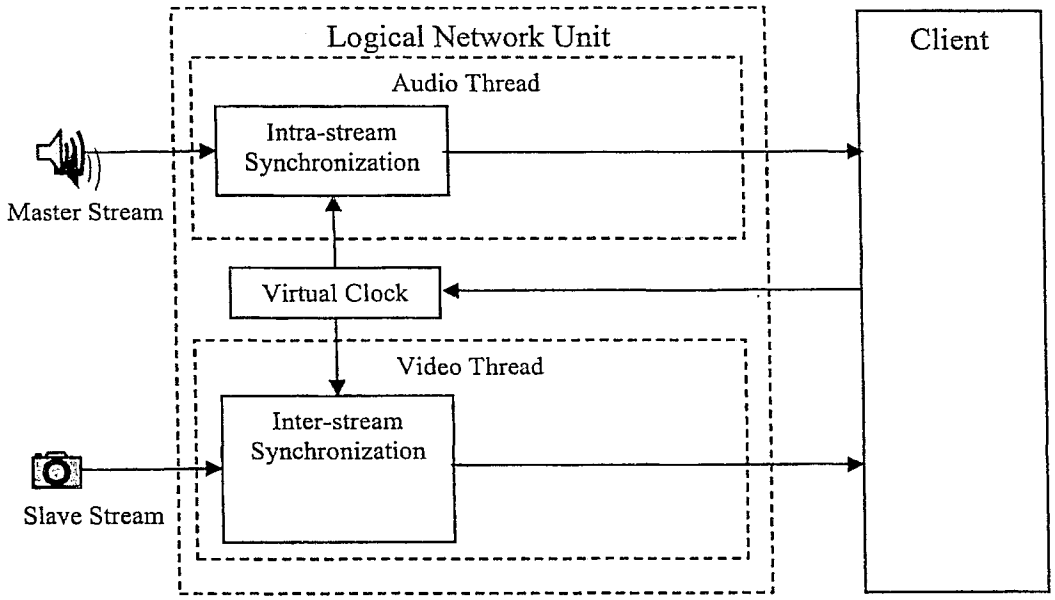


Figure 3.6 WAX Server Synchronization

Minimal time delay for presenting live data is desirable for reliability and safety of teleoperation. Therefore WAX attempts to use a small number of simple but effective methods to reduce the complexity and overhead of the stream delivery process.

## 3.3.2 Intra-Stream Synchronization

WAX does not take extra steps for intra-stream synchronization for video streams since the video frames are snapshots from the camera, taken at regular intervals, and these snapshots can be discarded at anytime for inter-stream synchronization to the master stream, which is the audio stream. However, since the video frames are taken at a regular interval, WAX indirectly provides a certain level of intra-stream synchronization.

For audio intra-stream synchronization, the WAX audio server tries to send each MU without discarding any, achieved by using three methods:

- Priority-based transmission (section 2.9.2)

- Static transmission rate adjustment (section 2.10.1)
- Static input rate adjustment (section 2.11.3)

For priority-based transmission, the audio server is given a higher execution and network socket priority than the video server. This ensures that when both video and audio MUs are ready to be sent, the audio MU is sent first.

Unlike video MUs which are made of individual snapshots, the audio MUs' length is the generation time (duration of recording) of an audio wave. For example, if the audio is recorded for 200 milliseconds then the audio MU length is 200ms. Since the audio sampling rate and bit rate (bits per second) are predefined for recording, the audio MU size can be easily calculated from the audio bit rate. If the audio MU length is long enough to compensate for the end-to-end delay for network traffic, then no audio MU needs to be skipped, because the next MU is always ready when the destination is playing the current MU. However, longer audio MU length suggests longer delay. Therefore, an appropriate audio MU length has to be used, which minimizes its impact on time delay for real-time operation. To reduce computational complexity, WAX does not use dynamic transmission rate adjustment but keeps the audio length constant. A suitable audio MU length can be determined by experimentation.

When the network bandwidth prohibits all the audio MUs reaching the destination, such as when the network bandwidth is lower than the audio bit rate, the audio MU length can not recover the asynchrony. In this case, WAX simply sends the audio MUs as soon as possible using source preventive skipping. This happens naturally, since WAX transmits whichever audio MU is in the shared memory buffer. The shared memory buffer always contains the latest audio MU, since the old MU is always replaced by the newly captured MU.

To ensure the safety of teleoperation and to improve intra-stream synchronization for audio streams, WAX uses a very low audio bit rate for the WAX server machine. On most computer systems including the WAX server machine, the lowest sampling rate is 8 kilohertz and the lowest bits-per-sample is 8 bits. Therefore, WAX's bit rate is 8 kilobytes per second. This bit rate produces phone-quality audio. Hence it is sufficient for transmitting speech and other audio for teleoperation.

While it is possible to use dynamic input rate adjustment (section 2.11.3) to lower the hardware audio bit rate to compensate for the network delay, in a multi-client environment, a change in hardware bit rate would require all clients to use the new bit rate. This scenario suggests that high bandwidth clients will be affected by any low bandwidth client. Clearly this is undesirable and is prevented by this technique. Also, there is a hardware limit on what the audio bit rate can be.

### 3.3.3 Inter-Stream Synchronization

WAX eliminates inter-stream synchronization for the audio stream since it is the master stream. This helps reduce computational complexity and improve real-time streaming performance.

Video streams on the other hand, need inter-stream synchronization to temporally align with the audio stream (if it exists). To achieve inter-stream synchronization, the video thread keeps a virtual clock between the video and audio streams. It calculates the inter-stream synchronization error by comparing the timestamps of MUs from both streams, and estimates the network delay of the sending process. The inter-stream synchronization involves the following methods:

- Stream MU length selection (Transmission rate adjustment)
- Buffering time adjustment with network delay estimation (section 2.9.6)
- Adaptive source skipping (section 2.11.1)
- Media Scaling (section 2.11.7)

#### 3.3.3.1 Stream MU Length Selection

The video and audio MU lengths have to be carefully selected so they can be temporally aligned with each other. Therefore WAX's audio MU length is a multiple of the video MU length. For example, if the audio MU length is 200 milliseconds (ms) then video MU length can be 200 ms, 100 ms, 50 ms, and so forth.

WAX's CCD camera is capable of capturing up to 10 frames per second, allowing WAX's video MU length (video capturing interval) to be 100 milliseconds. It is possible to use better cameras such as video camcorders to provide higher frame rate. However, the slowest process during stream processing is video compression, not capturing. Compression has to be done since raw image size can take hundreds of kilobytes to a few megabytes depending on resolution, while compressed JPEGs can be less than 10 kilobytes depending on the quality tolerances of the application. Also JPEG images are very useful for the web. The video MU length must be equal or longer than video compression time, otherwise some snapshots have to be discarded and server resources are wasted.

Since the video MU length is 100 ms, the audio MU can be a multiple of 100 ms; the audio MU must be long enough to compensate for the end-to-end network delay, and be able to provide continuous audio playback.

### 3.3.3.2 Buffering Time Adjustment with Network Delay Estimation

Since the video MU length is 100ms, as long as this length can compensate for the end-to-end network delay--i.e. the client is able to receive the next frame completely before the next frame display interval--then intra-stream of the video stream (frame rate) at the destination can be preserved. If the intra-stream for both video and audio streams can be maintained at the destination, and their starting points are temporally aligned (i.e. using a virtual clock), then the video and audio streams can be synchronized at the destination.

When an MU is captured from the hardware device, it is first stored in memory or other storage devices, and then it is processed into an appropriate format to use, such as the JPEG format for images and WAV format for audio. The system then starts capturing the next MU. The processing of the MU can be done by another thread. However, only the saved MU in memory is ready to be sent to the client. The client cannot access the MU while it is being captured or processed. Therefore, the client can only retrieve the previously captured MU from the server.

When the WAX server receives a connection from the client requesting both video and audio, the WAX server needs to perform the following tasks:

- Delay the transmission of video MU to compensate for the late Audio MU.
- Determine the buffer length for video MUs according to the network bandwidth available.

To temporally align the starting points of the first audio MU with the corresponding video MUs, the video MUs must be delayed. The recording of the audio wave is longer than capturing a video snapshot in nature; sending the video MU must be delayed to match the audio MU.

Once the starting points of video and audio streams are temporally aligned, The WAX server needs to cache the video MUs corresponding to the audio MU within the same time frame. For example, if an audio MU is 300 ms and video MU is 100ms, then two video MUs need to be cached. The caching is needed because WAX's video server only keeps one (latest) video MU in shared memory. Each video thread has its own video buffer for its client, which allows it to be independent from other video threads.

When the WAX client connects to the WAX server and requests both video and audio, the WAX server sends the previous video and audio MUs which are currently in the buffer., namely, $A_1$ (audio MU 1) and $V_2$ (video MU 2) in Figure 3.7. However, the WAX server must delay sending of $V_2$ by $T_p$ in order to synchronize with the audio MU $A_1$.

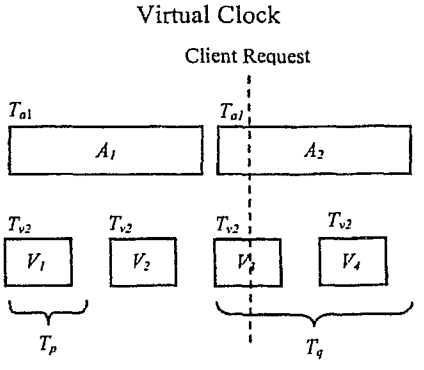


Figure 3.7 MU alignment during client request

$T_p$ can be obtained by the difference in timestamp $T_{a1}$ and $T_{v2}$:

$$T_p = T_{v2} - T_{a1} \qquad (3.1)$$

While the WAX video server thread is delaying for $T_p$ before sending the video MU, it has to cache the video MUs in its own buffer for up to $T_q$ because the WAX video server's main thread only keeps one copy of the latest MU.

To determine the network bandwidth, instead of waiting for $T_p$ interval and then sending $V_2$, WAX video thread sends $V_2$ first and then waits for $T_p$. This way the WAX video thread is able to obtain the network bandwidth by measuring the transfer speed of $V_2$. During wait time the WAX video thread is able to calculate and allocate the proper size of buffer for video images.

Once the transfer speed of $V_2$ is obtained, the number of video frames to cache can be obtained by dividing audio MU length by the transfer speed of $V_2$. Suppose $T_r$ represents the transfer time of $V_2$ and $N_v$ is the number of video frame to buffer,

If $T_r < V_2$, then

$$N_v = A/V_2 - 1 \qquad (3.2)$$

else if $A > T_r > V_v$, then

$$V_2 = A/T_r - 1 \qquad (3.3)$$

else when $T_r >= A$, then

$$N_v = 0 \qquad (3.4)$$

A is the audio MU length and it is a constant. If $T_r$ is less than $V_2$, it means the WAX server is able to finish sending an image before the next snapshot. Therefore, WAX can allocate the maximum buffer for the full frame rate. There is no need to allocate more buffers

than the possible frame rate. This is an ideal state, for example, if the audio MU is 500 ms and the video MU is 100 ms, then $V_2$ is 4.

If $T_r$ is more than $V_2$ but less than the audio MU, then the buffer length needs to be adjusted according to the network bandwidth.

If $T_r$ is equal to the audio MU, then no buffering is needed. If $T_r$ is more than the audio MU length--meaning the network bandwidth is simply insufficient--then the WAX video server enters a fail-safe mode for real-time teleoperation, which just sends the video image as soon as possible with no attempt at synchronization.

### 3.3.4 Adaptive Source Skipping

When bandwidth is insufficient for sending video images within the time frame, WAX uses adaptive source skipping to maintain a certain degree of synchrony even under very low network bandwidth conditions. The WAX video server's main thread only keeps the latest video MU in the buffer, and the WAX video server's client thread only updates the MU in its buffer when one of the MUs in the buffer finish sending. This way the MUs in the buffer are kept as recent as possible. Consider two scenarios:

- When $A > T_r > V_2$, and
- When $T_r >= A$

When $T_r$ is within the range of audio MU length and video MU length, WAX's video thread allocated the appropriate number of buffers. The number of buffers also indirectly affects the frequency which the images in the buffer are updated. For example, if $T_r$ is 200 ms and two buffers are allocated for 400 ms time frame, then the image is updated every 200 ms because the WAX video client only updates the image when one MU in the buffer is finished

sending and the slot is empty. In other words, the WAX server adaptively changes the frame rate of the video to match the available network bandwidth.

When $T_r$ is equal or higher than the audio MU, then the WAX video server just sends the latest video MU from shared memory as soon as the previous MU is finished sending. Since the video MU to be sent is always the latest, a certain degree of synchrony can be achieved at the client side, only limited by the network delay, which is unavoidable.

### 3.3.5 Media Scaling

To improve the real-time performance and to reduce the bandwidth requirement of the video stream, WAX uses media scaling to choose the size and quality of the shared JPEG image in shared memory to compensate for any network delay. However, this process is not dynamic only a one time process. Dynamic scaling is not ideal for a multi-client environment because each client experiences different network conditions. Therefore, dynamic scaling for each client requires recompression. Recompression is a very CPU and time consuming process. For real-time stream, recompression may take a longer time than simply sending a low quality image.

## 3.4 Destination Synchronization Control

The synchronization control at the destination is only used to preserve the synchronization done at the source. For simplicity, the destination synchronization control is done in one program. WAX's client is written in Java to provide platform independence. The Java API also provides standard routines for displaying JPEG images and playing back AU sound. AU audio is Sun's proprietary audio format. It is convenient to integrate all the

processes into one program. The integration is also desirable for synchronization, since the WAX client can have a total control over the video and audio threads. There is no need for any inter-process communication (IPC).

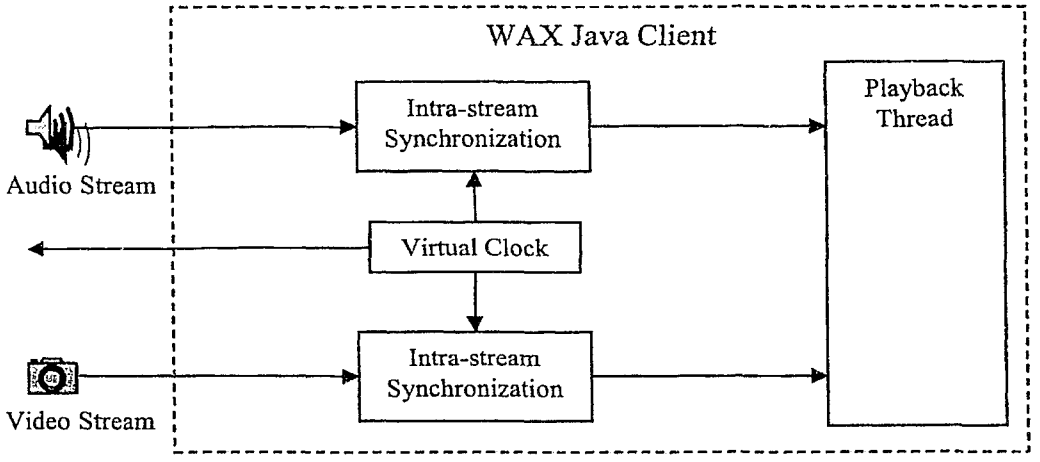


Figure 3.8 WAX Client Synchronization

## 3.4.1 Intra-Stream Synchronization

The WAX Java client performs intra-stream synchronization for both video and audio streams. Intra-stream synchronization is needed at the destination to overcome network jitter and delay during the transferring of MUs, as well as to maintain the playback rate of MUs. For example, if the video frame rate is 10 fps, then WAX client should playback the video MUs every 100 ms.

When the WAX client grabs the first video frame (MU), the frame is immediately displayed. It also sets the display time of the first frame as the starting point for the video stream on the virtual clock. Since the WAX webcam can only capture a maximum of 10 fps, the WAX Java client has 100 ms to grab the next image. This should not be a problem if the video image size is small enough. For example, on the 10 megabits per second (Mbps)

network, it is able to transfer a 5 kilobyte image in 4 ms if the network connection is not saturated. Note that JPEG file size is not constant, even if the image dimension and percentage of quality remains the same, since the size depends on the complexity of the image. However, the difference is only a few bytes and it does not affect the overall frame rate of the client, since the WAX client has enough time to obtain the image within the time frame.

Audio is a continuous wave therefore it does not have a well-defined frame interval unlike video. However, the audio stream comes in one audio MU at a time. The audio MU length is predefined at the WAX server so the WAX client is able to retrieve the MU completely before its playback time. In other words, the audio MU length is designed to overcome the end-to-end network delay. Therefore, to take full advantage of this buffering and to achieve a continuous audio playback, the WAX client also needs a buffer which is large enough to hold one audio MU. Note that this buffer is not used to delay the playback of the audio MU, but to be able to retrieve one full audio MU at a time. To ensure near real-time performance, the audio MU is played as soon as it is retrieved. To achieve continuous playback, the WAX client parallelizes operations by using two threads for the audio stream: a network thread and a playback thread. The playback thread plays back the current audio MU while the network thread retrieves the next audio MU to be played.

If the network bandwidth is higher than the transmission rate of full audio stream with the video stream, then continuous audio playback can be achieved. Unlike the video MU, the size of the audio MU is fixed since it does not use compression and the size is determined by its predefined bit rates, for example, the WAX server uses the lowest hardware setting available which is:

- Sample rate: 8 KHz
- Bits-per-sample: 8-bits (16-bit hardware bit rate compressed)
- Channels: Mono (one channel)

Java's AU audio format supports A-law and U-law compression, which compresses 16-bit audio into 8-bit by logarithmic quantization. Many sound cards also support hardware A-law and U-law compression. One byte per sample represents 8-bits. With a sample rate of 8 KHz, there are 8000 samples per second per channel. Since there is only one channel, WAX's audio bandwidth is 8 kilobytes per second, or 64 kilobits per second. In comparison, a 10 Mbps network is able to provide 10,000 kilobits per second, and takes about 6.4 ms to transfer one second of audio. Since the audio MU has to be a multiple of 100 ms, the WAX client should be able to obtain the audio MU in time for most cases.

### 3.4.2 Inter-Stream Synchronization

Most of the inter-stream synchronization control operations are conducted on the WAX server side. Therefore, the WAX client does not do any inter-stream synchronization, except to start playing the first MU from each stream as soon as possible, and to keep the temporal alignment between streams by maintaining intra-stream synchronization for each stream. It is because the WAX video server uses adaptive source skipping, causing the video MUs to be skipped and hence is missing from the video stream when the network bandwidth is insufficient. On the other hand, the adaptive source skipping helps the WAX client obtain the most recent video and audio MUs. As a result, even under limited bandwidth, a reasonable degree of inter-stream synchronization can be maintained. This approach improves the real-time performance and safety of teleoperation.

## 3.5 Analysis

The WAX synchronization algorithm uses many synchronization techniques to allow multiple streams to be synchronized based on the temporal alignment of their playback rate, at the same time it maintains a reasonable degree of real-time performance, since WAX is a real-time teleoperated system and safety is still top priority.

To avoid skipping of any MUs, WAX needs a minimum network bandwidth, a limit that can be calculated. WAX uses its webcam's maximum frame rate, which is 10 fps. Using media scaling, the image file size (video MU size) can be adjusted. For WAX, with an average file size of 8 k, the minimum network bandwidth for the WAX video server is 80 kilobytes per second, or 640 kilobits per second. With a 10 Mbps network connection, this should not be a problem. For a 64 Kbps dial-up connection, the client can only retrieve one frame every two seconds. With WAX adaptive source skipping, the client on dial-up can still receive the most up-to-date image.

The WAX audio server records sound at 8 KHz, 8-bit mono, allowing the minimum network bandwidth for the WAX audio server to be 8 kilobytes per second, or 64 kilobits per second, the same bandwidth as a 64 Kbps dial-up connection.

Therefore, to receive a continuous audio playback with full video frame rate, the client needs a minimum network connection of 88 kilobytes per second, or 704 kilobits per second. Otherwise, the WAX server drops old video frames to adapt to the network conditions.

It is difficult to remotely control WAX over a dial-up connection due to limited bandwidth. If the user has two dial-up modems installed on one computer, then the network

bandwidth is capable of providing a continuous audio playback with a frame rate of 1-3 fps. Users with just one dial-up connection should be able to maintain a continuous audio playback with slow image update. Unlike JPEG compression, U-LAW audio format is compressible to an even smaller size using hardware compression found in dial-up modems, leaving the remaining bandwidth for images.

Otherwise, WAX operates in fail-safe mode, which sends the latest video and audio MUs as soon as possible, and relies on adaptive source skipping to maintain a reasonable degree of synchronization. This happens when there is less than one frame per second, or the end-to-end delay of the audio stream is longer than the audio MU. Note that depending on the tasks, the user can also reduce the number of media streams transmitted (section 2.10.2) to improve the real-time performance of any particular stream.

# 4. WAX System Implementation

## 4.1 WAX Hardware Design

WAX's hardware is made of two parts: the actual robot and a server computer. The robot is composed of the following main parts.

- A camera
- A microphone
- A transmitter
- A receiver
- A micro-controller
- Wired or wireless links
- A battery pack
- A Chassis, and
- At least two motors and two wheels for mobility

WAX's chassis is made out of a wood-based material called medium density fiber (MDF). It is lightweight compared to metal and very easy to custom mill into a variety of shapes and forms.

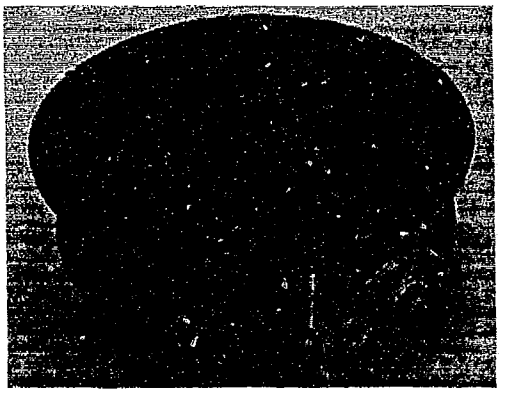

Figure 4.9 Blank robot chassis made of medium density fiber

WAX's MDF components, including the two level plates and wheels, are all manufactured by cutting large pieces of MDF using a computer controlled milling machine. The wheels are bolted separately to a 50-tooth gear and this assembly is mounted onto the end of a metal drive shaft.

WAX is driven and differentially steered by two 5V DC gearhead motors made by Sanwa Electric. The motor uses a 20-tooth pinion gear to drive the output gear in the gearbox.

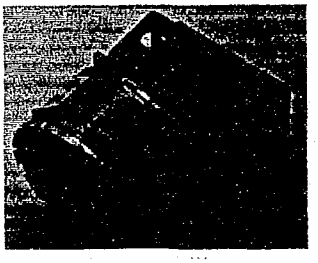

Figure 4.10 WAX gearhead motor

The motors are mounted at the bottom of the chassis. WAX also includes a small wheel with a spring at the end of the chassis to provide suspension to keep WAX in balance. A plastic bumper is used to provide protection at the front from collisions.

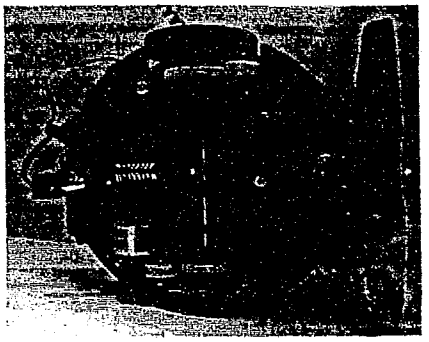

Figure 4.11 Bottom view of WAX

The camera and the microphone are used to capture video and audio signals. To achieve acceptable image quality, a Charged-Coupled Device (CCD) camera was used. The camera and the microphone were connected to a wireless transmitter. A wired transmitter can also be used.

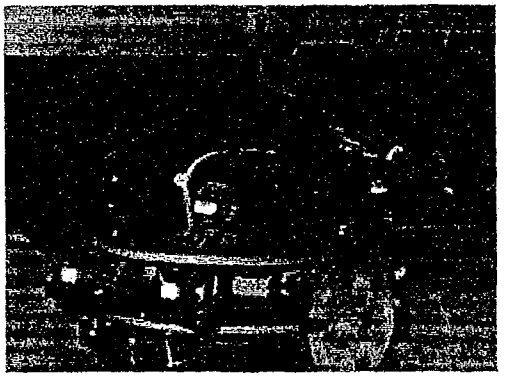

Figure 4.12 WAX

The transmitter used is a Radio Shack 2.4GHz wireless video/audio transmitter. It is used to transmit video and sound from the robot to the robot server. WAX uses a wire to connect the robot's micro-controller to the robot server. The wire is used to receive commands from the server computer to the robot, which are sent by the operator. A wireless transceiver can be used to replace the current transmitter and eliminate the wire. The battery pack is used to power the camera, the micro-controller, the transmitter as well as the power

motors for the wheels. WAX reuses the same wire for robot commands to obtain power and to recharge the battery pack.

The initial design of WAX uses a MC68HC11 Mini-board Microcontroller [57]. The microcontroller is used to control the two motors to drive the robot according to the operator's command. It can also perform more operations using its available input and output ports, such as pan and tilt of the camera, or activate a robotic arm.

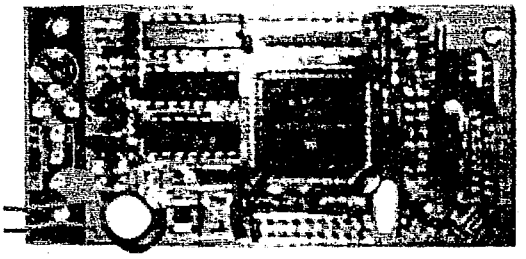

Figure 4.13 MC68HC11 Mini-board Microcontroller

WAX's Mini-board was later replaced by a successor MC68HC11-based board called the Handy Board [115]. Although the Mini-board was sufficient for this project, the Handy Board enables WAX to be used by various other projects that are currently being conducted in the lab.

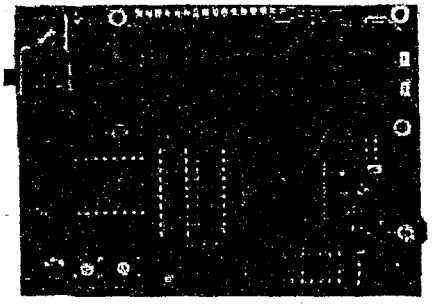

Figure 4.14 The Handy Board

The microcontroller is programmed using Interactive-C [56] [57]. Interactive-C is a commercial C compiler for Motorola 6811 based robots and embedded systems, which runs

on Windows, Mac, Linux, and other platforms. It is much easier to use than to program the microcontroller using assembly language.

## 4.2 Server Implementation

WAX's server is housed on a Pentium II 400Mhz machine running Slackware Linux. The CPU is much faster than the original MAX, which used a Pentium Classic 200Mhz machine. MAX is powerful enough to stream video with a hardware-limited frame rate without delay. Therefore, for WAX the CPU is not the bottleneck of the system, although faster is always better, especially for a multi-client server.

On the software side, WAX needs five programs:

- A web server
- An image grabber
- A video server
- A robot control server, and
- An audio server

The web server is used to allow the robot operator to visit WAX's home page, provide instructions and downloads for the Java robot control applet. The image grabber captures the images from the capture card and compresses them into JPEG format, while the video server serves the images to clients. The image grabber is kept separate from the video server to provide a layer of hardware abstraction. On the other hand, the audio server does both recording and sending of sound clips because the server can read audio wave data directly from the microphone.

The web server is the standard Apache server which comes with the Linux

distribution used for WAX. The video server, the robot server, and the sound server are programmed in C with UNIX IO and Networking API.

Unlike MAX which relies on the capture card's proprietary SDK (PMS-Grabber) for the implementation of the image grabber, WAX uses bttv [58] which is built into the Linux Kernel. Bttv is the Linux driver for TV and capture cards based on the bt848 and bt878, as well as other chipsets. These chips are being used on cards produced by major manufactures, such as ATI, IBM, Leadtek, Hauppauge, and Pinnacle. To capture the images, WAX uses a program called "webcam" from the package "xawtv". Webcam grabs the image from the capture card's video-in using the bttv driver and then saves the file on disk in jpg format. A ram disk is made under Linux to reduce the overhead of saving the image to hard disk. The image grabber is set to capture images at 10 fps because that is WAX's webcam maximum capture rate.

WAX video server is improved upon MAX's code. Instead of using a new port number for each new client, WAX uses just one port number for all clients, much like how other UNIX daemons are implemented. For each new client, WAX forks a new thread. Since the JPEG image size is always different due to the fluctuating complexity of the image, the video thread first sends the size of the file followed by the actual image. The new thread reads the image from the ram disk and sends it to the client whenever the client is ready. Depending on client's network connection, each thread is able to adapt to each client's bandwidth restrictions and sends the images at the client's speed.

To lower the bandwidth requirement of video streaming, WAX uses media scaling to lower the size and quality of the JPEG image. The image dimension is set to be 166 pixels in width and 129 pixels in height. The image quality is 75%. This yields an approximate file

size of 8 kilobytes per image.

The WAX robot control server's task is simply to forward the operator's commands to the robot through the computer's COM port for transmission. To add functionalities to the robot, the new program can be uploaded to WAX's on-board microcontroller.

The WAX audio server uses the Open Sound System (OSS) [116] sound driver, which is built into the Linux Kernel, to capture sound. Like bttv, OSS provides the hardware abstraction layer for sound programming under Linux. In other words, if the sound card works under Linux, it will work with the WAX audio server.

To reduce CPU processing, and bandwidth requirements while maintaining acceptable quality, the audio server uses a common recording setting supported by the sound card hardware. The audio is recorded in mu-law (also known as u-law) compression with a sample rate of 8 KHz 8-bit mono. Mu-law is a logarithmic quantization that takes linear samples of 14 to 16 bits and quantizes them to 8 bits. It produces good sound quality for human hearing while compromising audio quality in higher ranges. It is a common encoding which is used in many telecommunication devices, such as telephones. Mu-law is simple and fast. Therefore, many sound cards have built-in hardware mu-law encoding, including WAX's sound card. The use of mu-law is also employed because it is the compression supported by Java Applets which only understand Sun's au audio format. Mu-law is the standard encoding for audio recording from the microphone on all Sun Microsystems hardware. There are many conversion tools available for writing au audio format. WAX uses the sndlib library for simplicity.

The WAX audio server utilizes UNIX's shared memory architecture to store the recorded sound clip. There is a dedicated thread to record the audio clip to the shared

memory. For each client connection, WAX spawns a new network thread which reads the audio clip from the shared memory and sends it to the client whenever it is ready. Unlike video images, the audio clip is at a fixed size. Because, regardless of the noise complexity, the audio is always sampled at 8 KHz 8 bit mono, therefore, the data rate is 8 KHz x 8-bit = 64 Kbps per second.

The WAX audio server is given a higher execution and network priority than the video sever. The UNIX "nice" command is used to start the servers with different priorities. The server can also change its priority using C function calls. For network priority, WAX uses the setsockopt() call to change the priority setting in the TCP flag of the video and audio server ports. The same result can be achieved by "iptables" or "ipchains" commands in Linux.

## 4.3 Client Implementation

The Java applet control client resides on WAX's home page, which the user can download from WAX's Apache web server. The Java applet includes three clients in one:

- The robot control client
- The video client, and
- The audio client.

The robot control client displays a control panel the same as MAX. The panel includes directional buttons, fine and gross control, user lists and a time counter. The user has 60 seconds to operate the robot, than the next user takes over, and the user is re-queued at the end of the queue. The robot control client sends control signals to the WAX robot server. The WAX robot server, in turn, sends the signal to the robot's on-board microcontroller.

The video client displays the frames obtained from WAX and displays them

according to the original frame rate. In other words, it sleeps until it is time to display the next image.

### 4.3.1 Java Audio

The WAX audio client is the most challenging part of the client implementation, because Java has very poor support for audio, even today's version of 1.4.2. The only audio interface available is AudioClip from the java.applet class. AudioClip has three methods: play(), loop(), and stop(). These methods allow AudioClip to play, loop and stop playing a static audio file. The problem with AudioClip is that it does not have support for streaming. The audio file has to be a static file, otherwise AudioClip waits forever for the streaming data to be downloaded before playing.

Since AudioClip allows Java to play audio, it should be possible to enable audio streaming through lower level programming. With some searching on the web, there is reference to the undocumented sun.audio.* package in Java which provides low level audio programming. However, sun.audio.* is not in the sources.zip source code archive provided by Sun. To find information regarding of sun.audio.*, the Java's classes.zip were decompressed and decompiled. The undocumented sun.audio.* package was then shown, which provides a low level programming interface to audio.

According to Sun's Java Programming Guideline [59], Programmers should only use java.* packages because they are the proposed platform independent reference library. The package sun.* is hardware specific to Sun. However, it was discovered that sun.audio.* is implemented on all Java-based web browsers, including browsers such as Internet Explorer and Netscape. This is because sun.audio.* is required for AudioClip.

The discovery of sun.audio.* has enabled WAX to maintain the same portability and platform-independence of the original MAX. No third-party media player is needed. Sun.audio.* also gives programmers greater control of audio streaming. It has many classes, including AudioDevice which interact directly with the underlying audio device.

For the Java applet interface, a new menu item is added to the control panel window. The menu allows the operator to enable and disable the audio at any time. This is useful when the operation does not require audio, or more bandwidth is desired for the video, or when another version of WAX is created without a microphone.

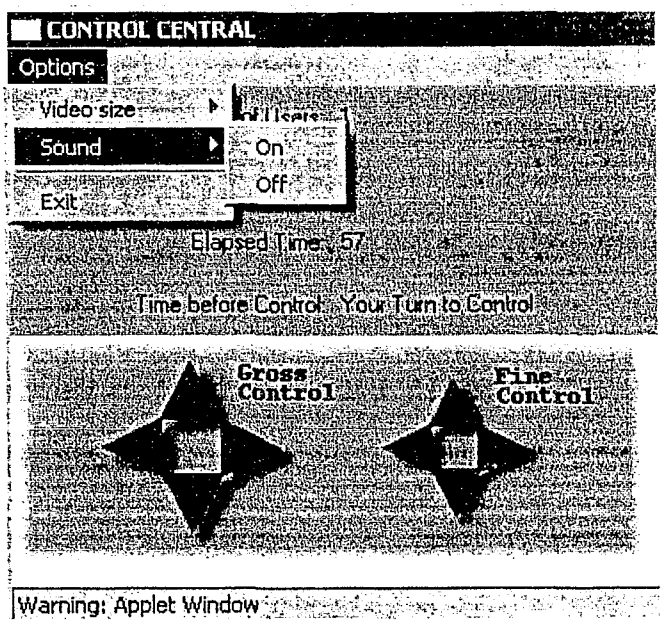


Figure 4.15 WAX's audio control menu

## 4.3.2 Audio Streaming

Another reason that AudioClip does not work for audio streaming is that it uses a buffer that is not a streaming buffer. AudioClip uses an array to store the audio data and then play it. An Array works well for video data. A common technique to smooth video playback is the

use of double-buffering. Double-buffering uses two buffers, while one is displaying the image on the screen, another one prepare the image off-screen. This improves the smoothness of the video.

However, double-buffering does not work very well for audio streams. Video is made of snapshots while audio is a continuous wave sampled at much higher rate. Even a small interruption in the continuity of the audio can easily be spotted. From WAX client experiments, with double-buffering, there is always a slight pause in audio during playback, because there is an unavoidable delay in buffer switching. A more elegant solution to this is to use a streaming buffer.

A streaming buffer acts like a queue. Data is retrieved from one side of the queue and exit at the opposite side of the queue. There is no defined beginning and end of the data. As long as the data can be received fast enough for data retrieval on the other side, the playback rate can be sustained.

The WAX client uses a streaming buffer to cache the incoming audio data while the audio thread is retrieving audio data from this buffer at its own rate for playback. Note that this buffer does not delay the audio playback, it only attempts to make the audio playback smoother by temporarily caching the incoming audio data when the audio thread cannot handle the data all at once.

Since this streaming buffer does not delay the audio playback, the length does not matter. For a fast connection, the audio playback thread can consume as much audio data as the WAX server can provide. For a slow connection, the audio playback thread always retrieves audio data faster than WAX server. Currently the buffer is set to be twice the size of audio MU, which gives enough room for the next audio MU.

# 5. WAX Experiments

## 5.1 Overview of Experiments

A series of experiments were conducted to examine WAX's ability to manage video and audio streaming for real-time teleoperation, and its synchronization performance. In order to evaluate video and audio synchronization, especially lip-sync, the WAX server is connected to the television output for TV programs with constant sound and human speech, because normally WAX's operating environment does not always have human actors. Testing was carried out in both a 10/100 Mbits/sec (Mbps) Ethernet Local Area Network (LAN) environment and over the Internet, with different hardware, operating systems and Java-enabled web browsers. Note that except for the bandwidth stress test, the Internet connection used for the tests is a nominal 5 Mbps Rogers high-speed extreme cable modem connection. The tests performed are:

- Intra-stream synchronization test: This checks if the WAX client can maintain the playback rate of video and audio streams. The main objective of this test is to determine the optimal audio stream MU size which can overcome the end-to-end delay for various types of network without introduce too much additional delay.
- Inter-stream synchronization test: This test analyzes the video and audio synchronization performance on different operating systems, and over various network configurations (LAN and Internet).
- Bandwidth stress test: This test analyses the degradation in real-time performance under various connections with limited bandwidth, including under wireless and dial-up

connections. The test is also used to test WAX's fail-safe feature and the level of real-time synchronization it can provide with adaptive source skipping.

- Portability and compatibility test: For this test, different operating systems and Java-enabled web browsers are used.

For each test, the WAX server and client are running with a debug output enabled. Depending on the test performed, different relevant information is collected. The data is either displayed on the console or saved to ram disk for analysis.

Despite the differences in each test, the WAX client is always in full operation mode and set to retrieve both video and audio streams. This provides a more accurate real-world performance measure, even if only a video or an audio stream is tested.

For each test, there are always human actors to verify the results.

## 5.2 Intra-Stream Synchronization

Intra-stream synchronization test is performed on both LAN and the Internet. For each environment, the test is separated into two sections. The first section tests the intra-stream synchronization of the video stream, and the other section checks the intra-stream synchronization of the audio stream.

To test intra-stream synchronization of the video stream, the WAX client records the following:

- The timestamp when it requests a frame
- The timestamp when it has finished receiving a frame
- The difference between the two timestamps

The difference between these two timestamps is the end-to-end delay of the video

MU transmission. Since the WAX video server streams at 10 fps, this end-to-end delay has to be less than 100 ms for a full frame rate. Otherwise, WAX adaptively changes the video frame rate according to the available bandwidth.

The test for intra-stream synchronization of the audio stream is not as straight forward as for the video stream with regard to the following:

- Audio data is a continuous wave, whereas video data is composed of individual snapshots. A pause in the audio stream is easier to detect and more critical than the video stream.
- The WAX client uses a stream (queue) buffer instead of an array buffer for the audio stream. The stream buffer makes smoother audio possible since it treats the incoming data as a real stream. Therefore, it does not mark when one audio MU finished and the next one began.
- The WAX client retrieves audio data from the stream buffer as soon as the playback thread requests it. Therefore, the size of the audio data in the stream buffer is different and independent of the audio MU length.
- When the "read" method of Java's stream class is blocked by an empty stream queue while reading, it simply returns successfully with the audio data read. If the next call to "read" is to an empty queue, the "read" method returns successfully with 0 bytes. The only exception event Java stream class uses is the IOException, which is thrown when disconnection occurs. Additionally, Java's audio class simply blocks and pauses the audio until more data is available instead of returning an error.

To overcome the limits imposed by the Java's streaming class, a new streaming class is derived from Java's default streaming class. This new class allows the monitoring of the

read method each time a new audio MU is required from the audio class. Therefore with this new class, the WAX client's audio thread is able to retrieve the same timestamp information as the video thread as well as other information:

- The audio length which is requested by Java's audio class
- The audio length which WAX audio thread is able to provide
- The time required for this transaction

The time for the transaction is the same as the time delay in the video thread, which can be measured through the following

- The timestamp when it requests a frame
- The timestamp when it has finished receiving a frame
- The difference between the two timestamps

Since the continuity of the audio stream is for human hearing, human actors are used to verify the pause in the audio.

Intra-stream synchronization for the audio stream is required for a continuous audio playback, and it is achieved by selecting an audio MU length which is long enough to compensate for the network delay. Otherwise empty gaps (audio pauses) would occur during audio playback.

This audio MU length also needs to be a multiple of the video MU length, so the temporal alignment of video and audio MUs can be maintained for inter-stream synchronization.

Since 100 ms is the minimum video and audio MU length, it is set as the starting point for audio MU length. The audio MU length is incremented by 100 ms for each test until the gap between the consecutive audio MUs disappears.

### 5.2.1 Local Area Network

In this section, the intra-stream synchronization for the video and audio streams is tested over a LAN.

#### 5.2.1.1 Video Stream

The first test is to run the WAX client directly on the server machine. This checks the processing delay of the video server. Running the client and the server on the same machine eliminates network delay. The WAX client should be able to get the video MUs instantly.

Since the WAX server is running Linux, the WAX client is running on Netscape 4.8 under the Linux X-Windows system. The time delay for receiving a video frame is displayed in the Java console.

Note: the X-Windows manager used for most of the tests is FVWM95. The windows title bar of FVWM95 looks much like Microsoft Windows. However, it is still under Linux. It is shown by the X icon in some of the windows. FVWM95 was used because it provides a friendlier working environment under a UNIX-based operating system.

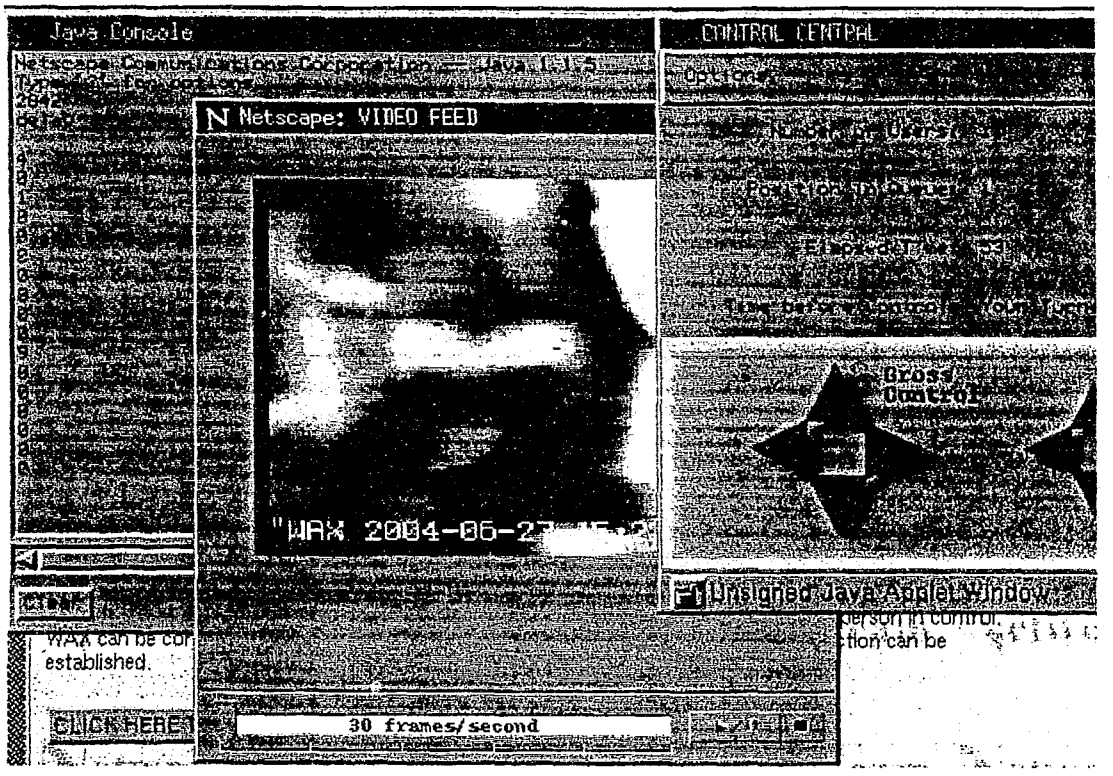


Figure 5.16 WAX video streaming on the same machine

In Figure 5.8, notice the first number in the Java console is 2842 ms. This is the time required for the WAX client to initiate a streaming connection with the WAX server. Since the test is running on the same machine, it means regardless of network condition, the initial connection would take at least 2-3 seconds to establish. This delay includes memory allocation, socket creation and function calls on the WAX client and server.

When the WAX client is running on the same host, on average the end-to-end delay for video MUs is less than 1 millisecond, with occasional spikes of up to 9 ms. These values are much lower than 100 ms. Therefore the WAX client should not have a problem maintaining full frame rate when running on the same host.

The second test is to run the WAX client from a machine on the Local Area Network

(LAN) with 10/100 Mbps Ethernet connection.

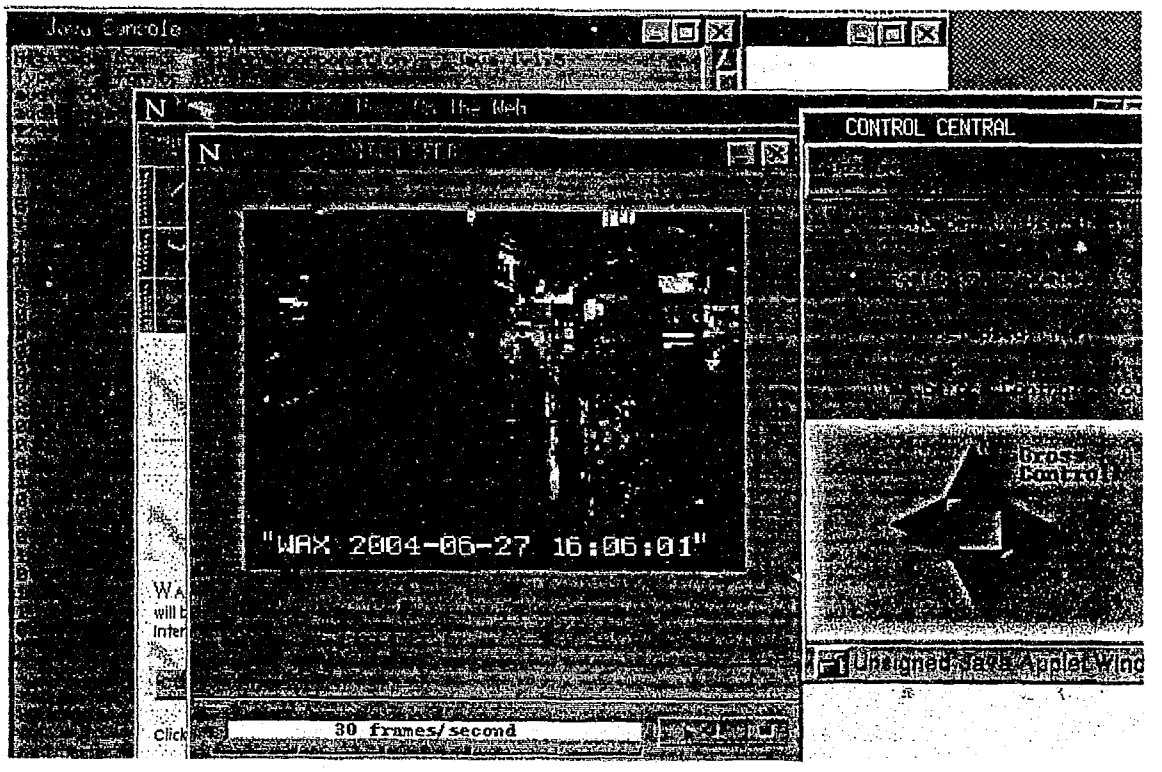


Figure 5.17 WAX video streaming on LAN

Since there is only one network hop (network switch) between the LAN machines, the WAX client is able to obtain the video MU as fast as if they were taken on the same machine. The network latency on the LAN is negligible.

The third test is to run WAX client on a machine from another building, which is connected to the same campus network but not on the same subnet (department). The Ryerson campus network connects computers from many departments together including computer science, electrical engineering, image arts, as well as student residents, etc.

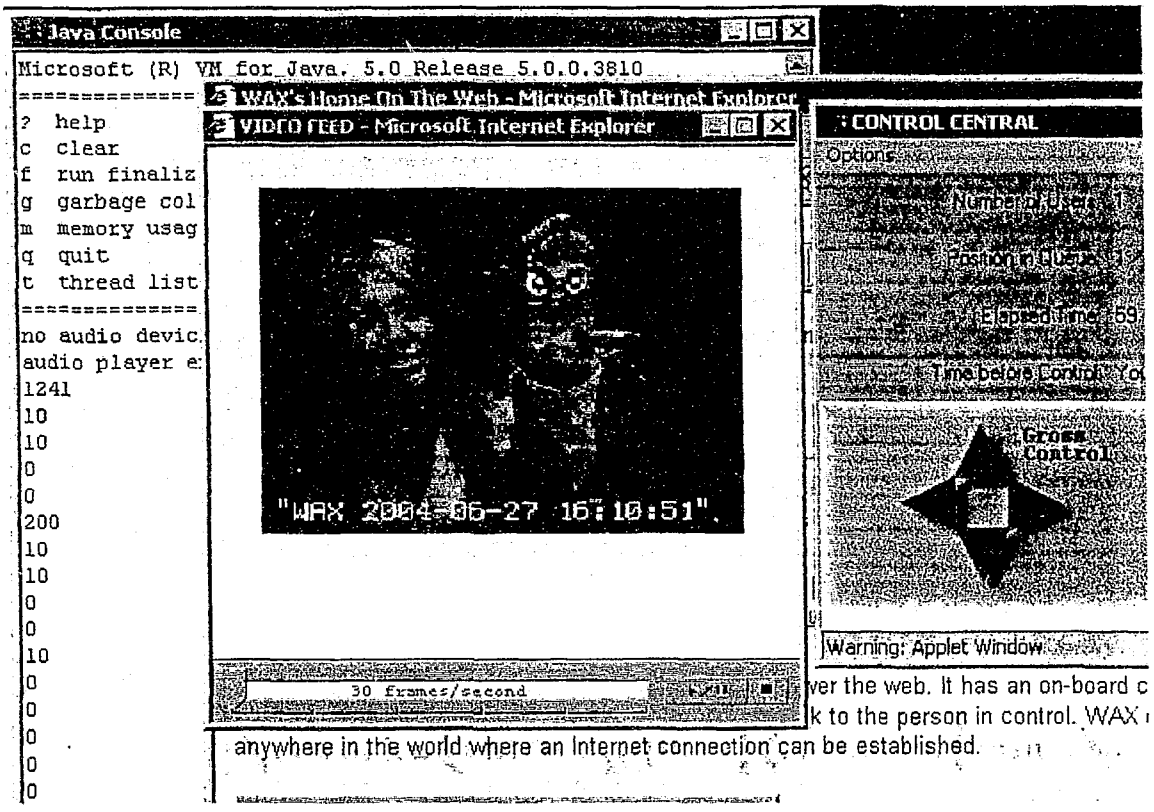


Figure 5.18 WAX video streaming on the same campus network

The third test is running on Microsoft Internet Explorer 5.0 with Microsoft Java VM under Windows 2000. On average the end-to-end delay is 0 or 10 ms. Since the network packets have to go through more network switches and these switches process packets from different networks on a first-come-first-served basis. Occasionally some of WAX's video MU packets can be delayed. As demonstrated in one of the test, the delay can be as long as 200 ms. In this case one video frame is dropped.

### 5.2.1.2 Audio Stream – 100 ms

The first test of audio streaming uses 100 ms of audio MU length. The client is running on the WAX server.

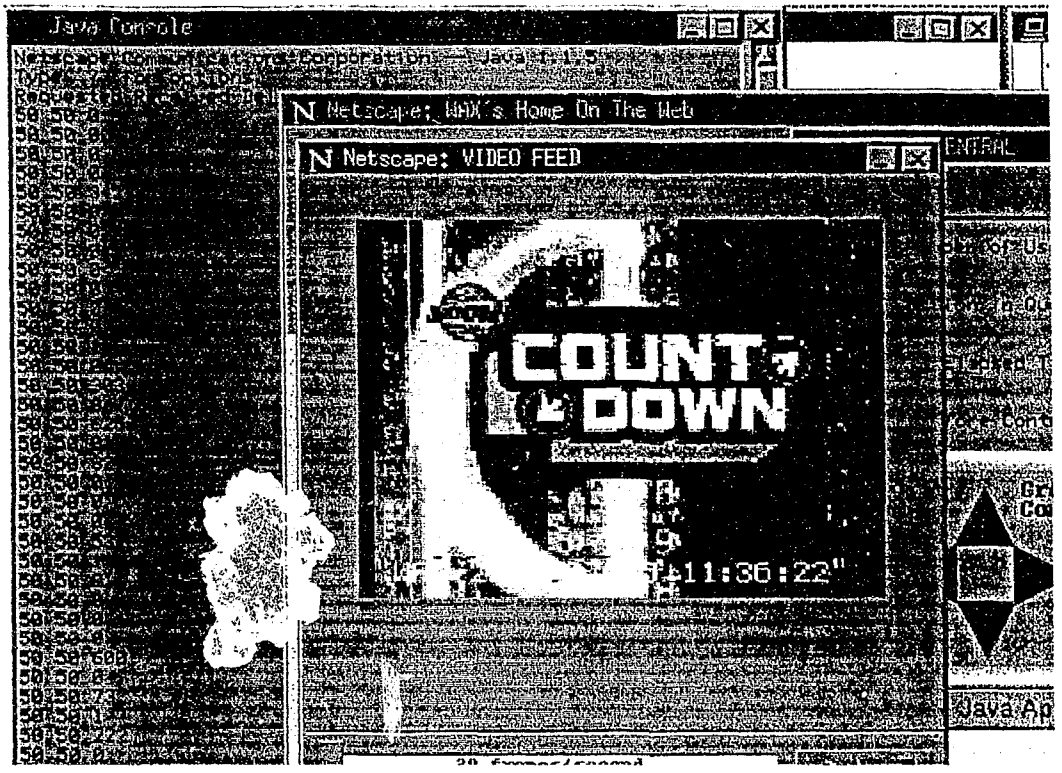


Figure 5.19 WAX audio streaming on the same host with 100 audio MU length

There are three numbers displayed on each line of the debug output:

- The audio length in ms which the audio playback thread requested
- The audio length in ms which the audio playback thread received from the audio network thread
- The time delay in ms for this whole receiving process, which must be less than the audio length requested by the playback thread (the first number) to produce an uninterrupted audio playback

As illustrated in Figure 5.11, even when the client is running on the same host. It can not maintain a smooth audio playback with 100 ms of audio MU length. Some delays are in the range of 300 to 600 ms. When the delay occurs, a pause can be heard in the audio

playback. The first number also shows that the internal buffer size for Java's audio engine is 50 ms.

The second test is performed on the same LAN machine as for the video stream test.


Figure 5.20 WAX audio streaming on a LAN machine with 100 audio MU length

There are about one to four pauses for every half second of audio. The pause can be as long as 800 ms.

The third test is performed on the same machine located on another building.

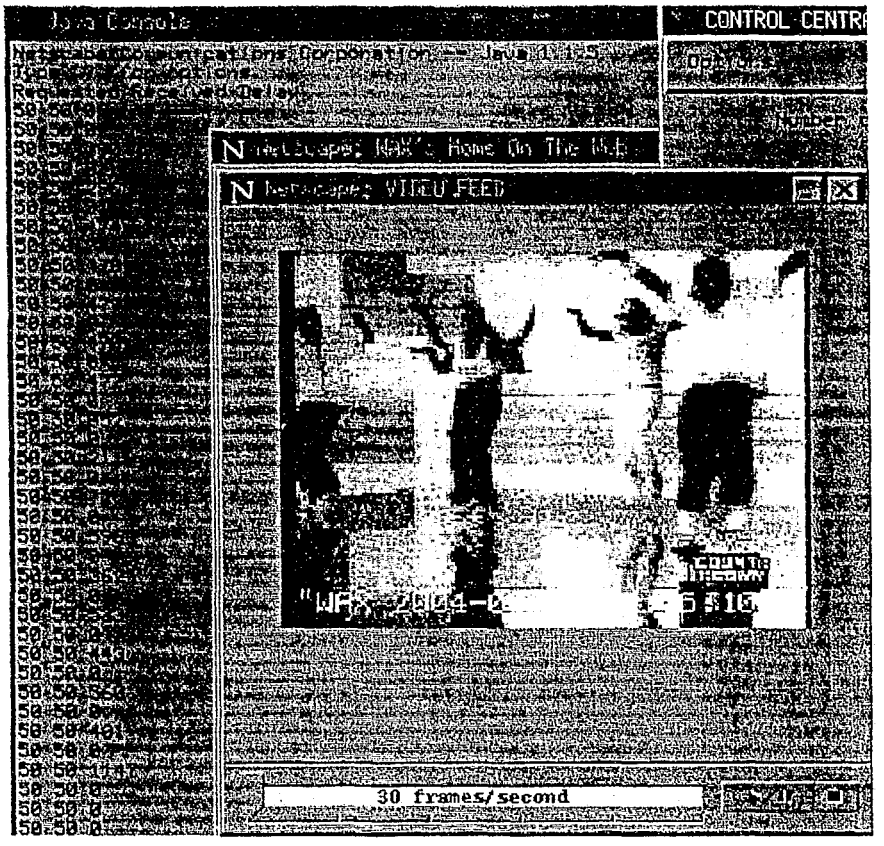


Figure 5.21 WAX audio streaming on a campus machine with 100 audio MU length

The frequency of pauses increases dramatically when the network packets have to travel a number of hops, almost every 100 ms there is a pause.

### 5.2.1.3 Audio Stream – 200 ms

The audio test suite is repeated for 200 ms audio MU.

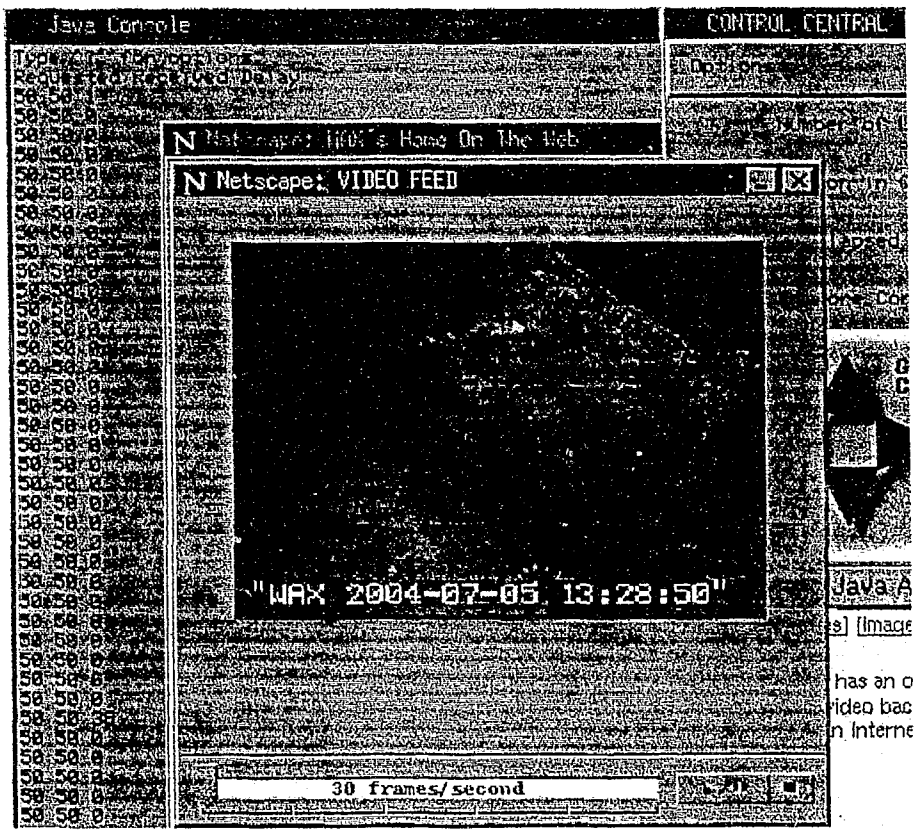


Figure 5.22 WAX audio streaming on the same machine with 200 audio MU length

With 200 audio MU length, the WAX client is able to sustain a continuous audio playback on the same machine. As illustrated in Figure 5.14, there is only one 38 ms delay for an audio MU after 1.5 seconds of playback. This delay is still less than 50 ms therefore Java's audio thread is able to recover from the delay quickly. Hence, this delay does not affect the continuity of the audio playback.

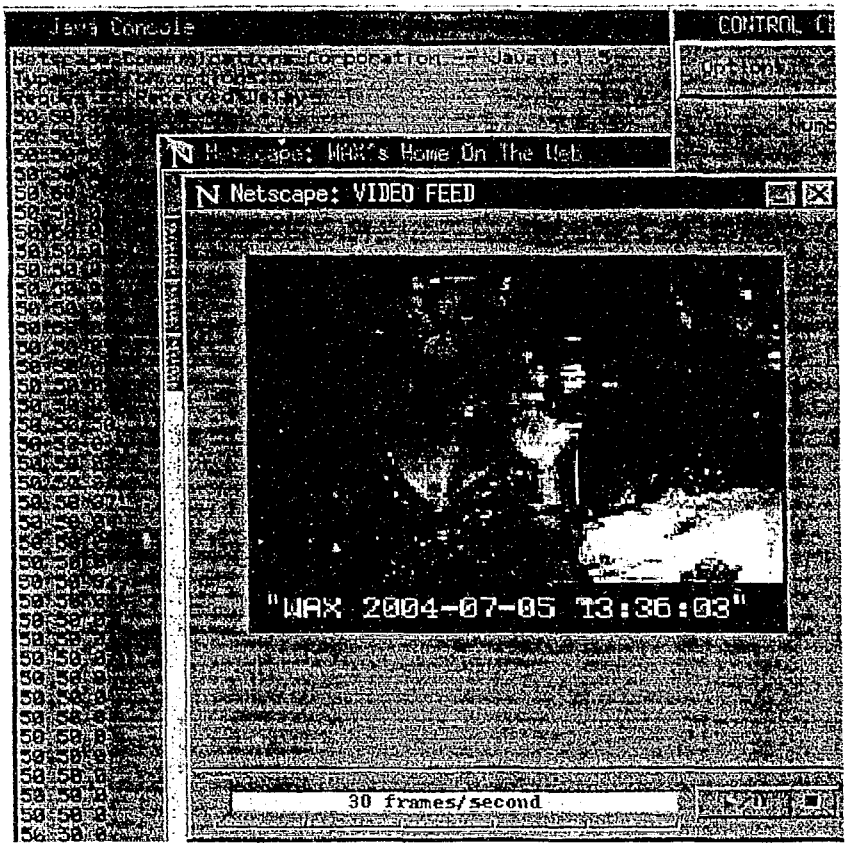


Figure 5.23 WAX audio streaming on a LAN machine with 200 audio MU length

When the test is performed on the LAN machine, on average the delay is 0 ms. However, there are one to three pauses for every one second, the length of the pause can be up to 400 ms.

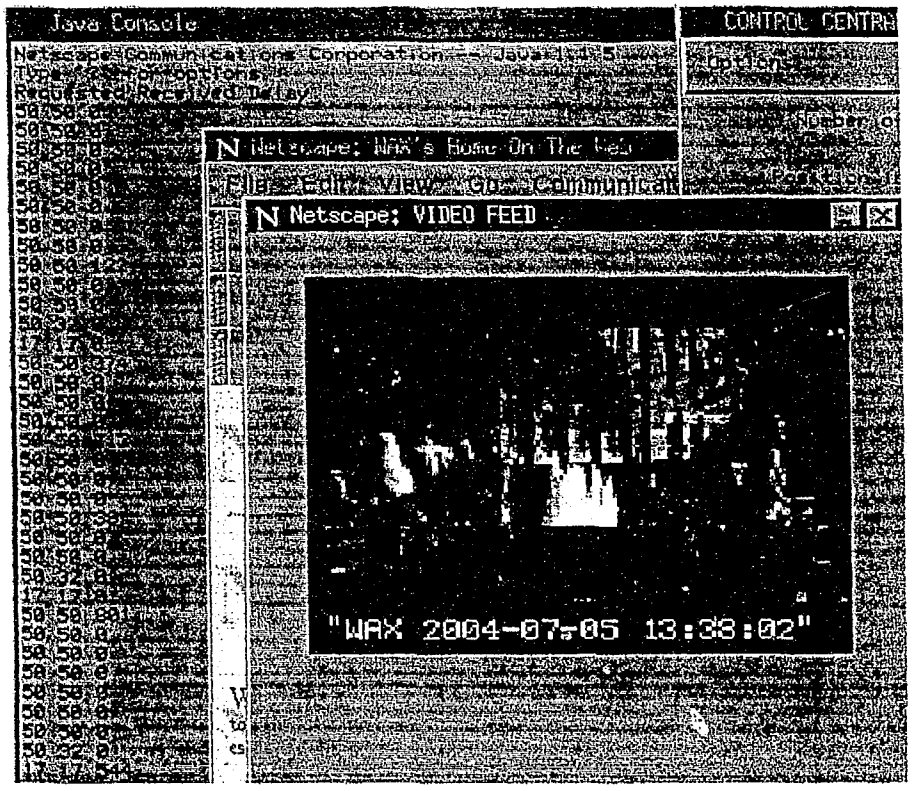


Figure 5.24 WAX audio streaming on a campus machine with 200 audio MU length

When the test is performed on the campus machine, pauses occur more frequently. On some occasions, although the playback thread is able to obtain the data without delay, the audio data received is less than the requested 50 ms. When this happens, the Java socket API attempts to recover the error by trying to get the remaining bits. As long as the Java socket thread can retrieve the remaining audio data without delay, it will not affect the overall smoothness of the audio playback.

For this experiment, there can be as many as 3 pauses in a second if the WAX client is run from a campus machine with 200 ms audio MU, and the pauses can be as long as 800 ms.

### 5.2.1.4 Audio Stream – 300 ms

The test on the same host is repeated with 300 ms audio MU.

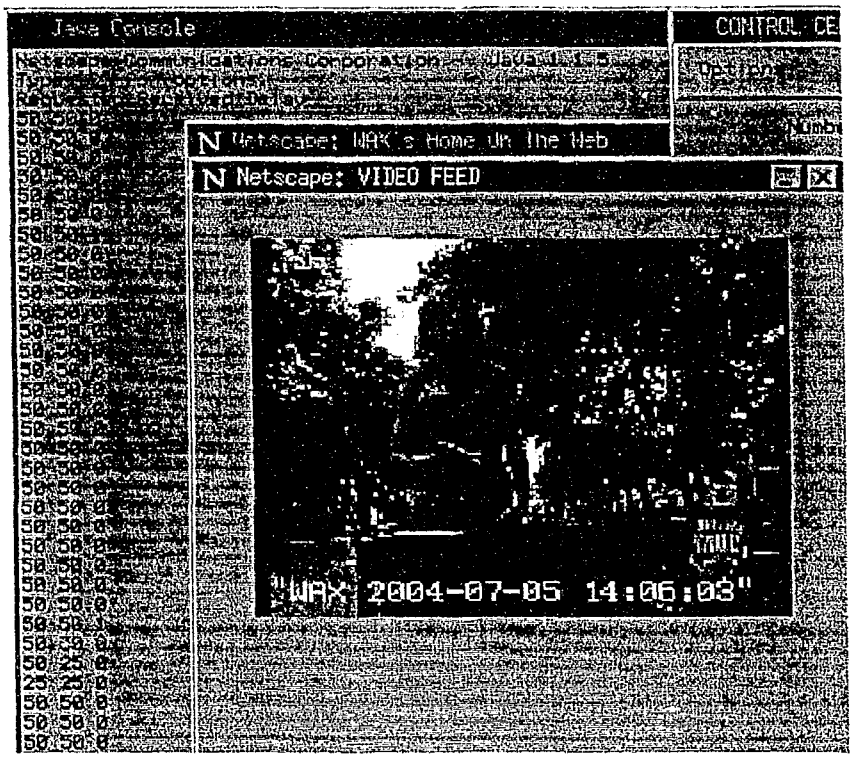


Figure 5.25 WAX audio streaming on the server machine with 300 ms audio MU length

The WAX client has no problem maintaining a continuous audio playback with 300 ms audio MU on the same host. There is no delay in getting the audio data from the network thread.

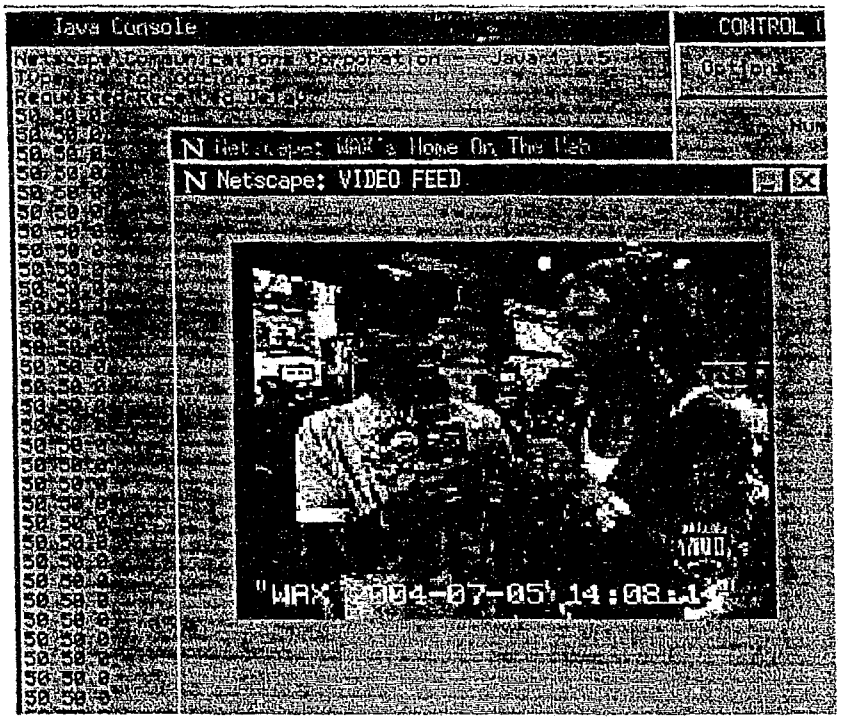


Figure 5.26 WAX audio streaming on a LAN machine with 300 ms audio MU length

Again, the WAX client is able to maintain a smoother audio playback on a LAN machine with 300 ms audio MU.

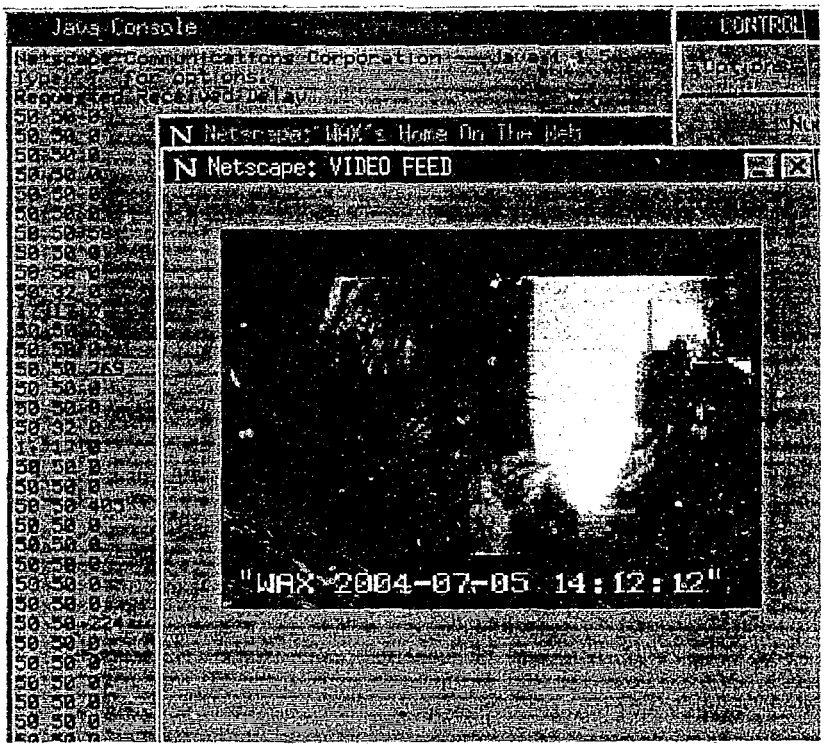


Figure 5.27 WAX audio streaming on a campus machine with 300 ms audio MU length

However, when the WAX client ran the same test on a campus machine, the pauses still exist during playback. For each second, there can be as many as five pauses. The pause can be as long as 400 ms.

### 5.2.1.5 Audio Stream – 400 ms

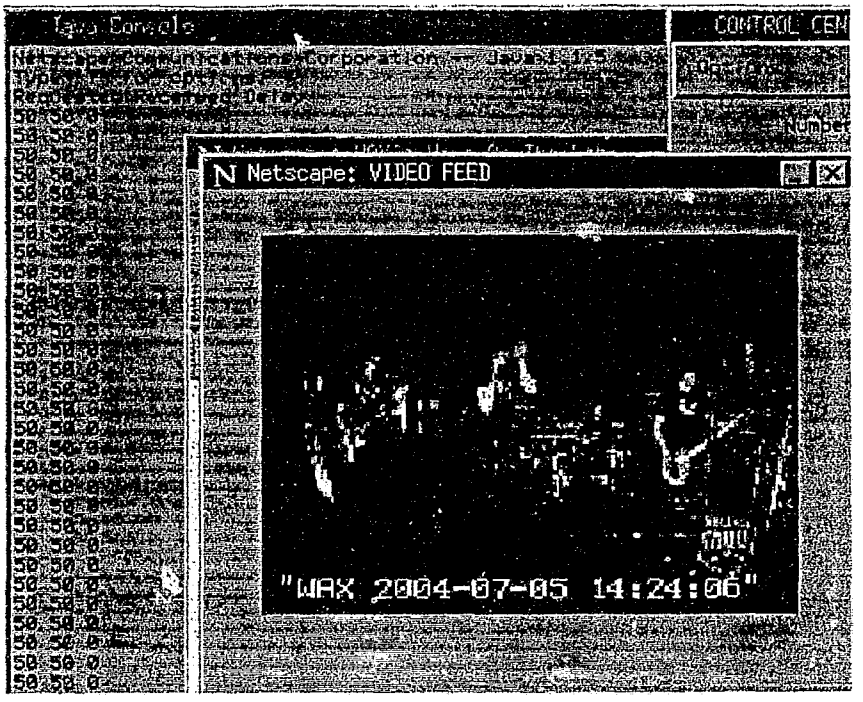


Figure 5.28 WAX audio streaming on the server machine with 400 ms audio MU length

The WAX client can achieve smooth audio playback on the same host with 400 ms audio MU.

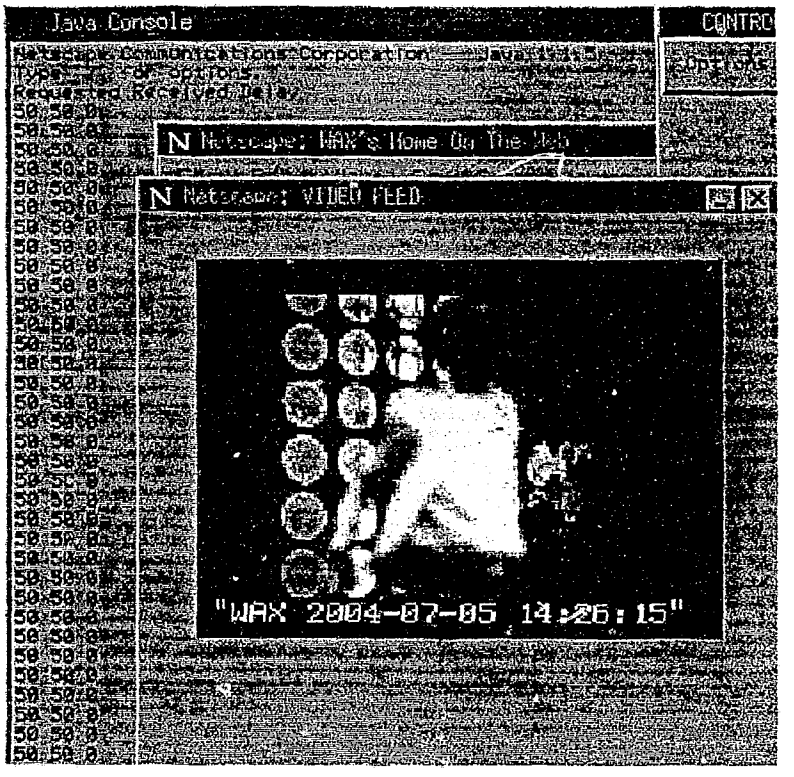


Figure 5.29 WAX audio streaming on a LAN machine with 400 ms audio MU length

Same as with 300 ms, The WAX client is able to accomplish a continuous audio playback with 400 ms audio MU.

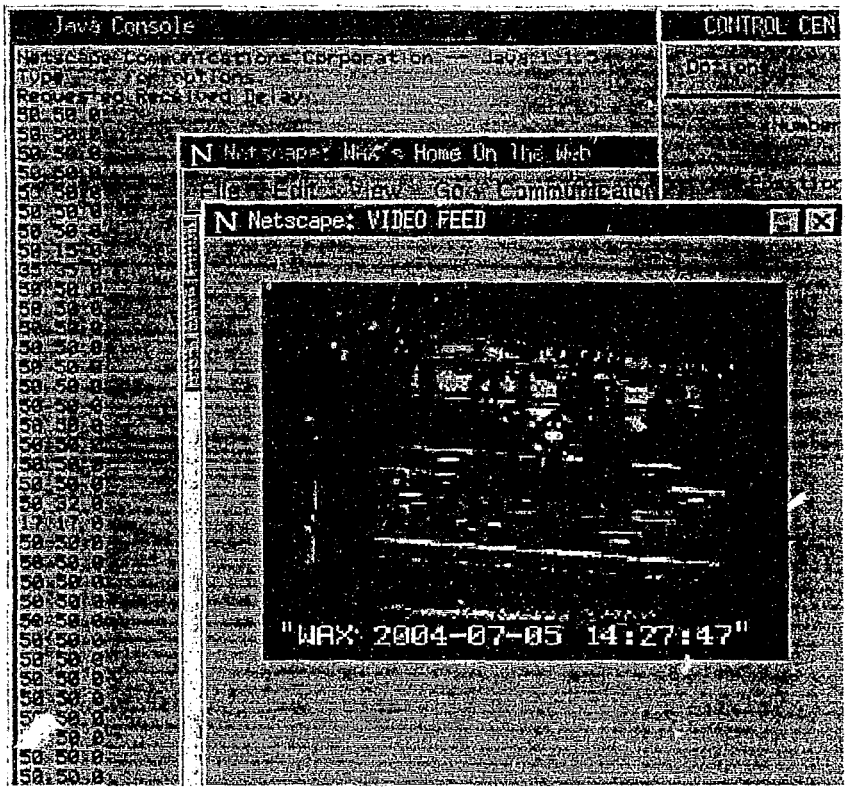


Figure 5.30 WAX audio streaming on a campus machine with 400 ms audio MU length

As illustrated in Figure 5.22, there is no delay in getting the audio data when the WAX client was running from a campus machine with 400 ms audio MU. However, occasionally the audio thread cannot get a full audio MU. This only happens no more than once every second.

Since the WAX client is able to maintain a smooth audio playback on the server machine, a LAN machine and a campus machine with 400 ms audio MU. There is no need to continue this test for local network machines with longer audio MUs.

To study how various audio MU lengths affect the continuity of the audio stream, graphs are plotted to illustrate the relationship between the audio MU length and the time delay in the audio.

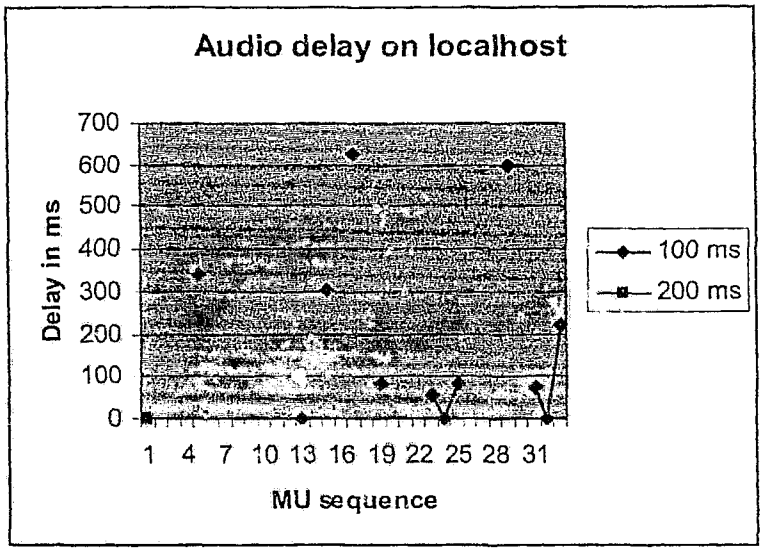


Figure 5.31 Audio delay on localhost

As illustrated in Figure 5.31, even on the same machine, a continuous audio stream cannot be maintained with 100 ms audio MU length, but with 200 ms.

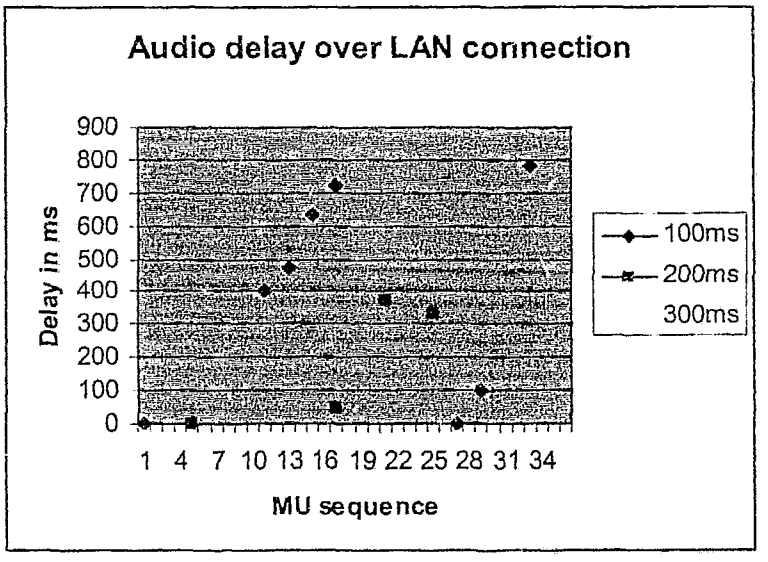


Figure 5.32 Audio delay over the LAN connection

With a LAN connection, 200 ms audio MU can reduce approximately 50% of the time delay, and 300 ms audio MU provides no time delay.

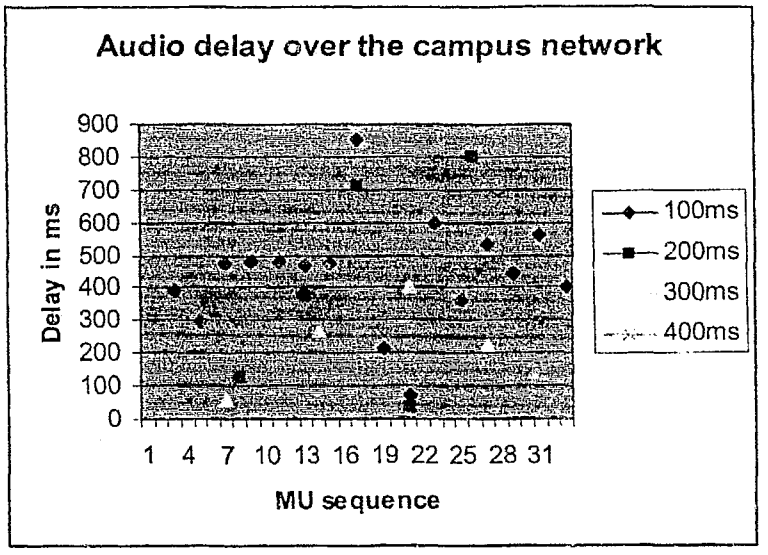


Figure 5.33 Audio delay over the campus network

When operating the WAX client over the campus network, the delay happens more frequently with 100 ms audio MU. 200 ms audio MU can reduce the occurrence of the delays to half. However, the duration of the delay can still be as long as with 100 ms. With 300 ms, the duration of the delays is less than 500 ms, and 400 ms can guarantee the continuity of the audio stream.

## 5.2.2 The Internet

For this test, the client's Internet Service Provider (ISP) is Rogers Communications. The connection is 5 Mbps high-speed cable modem connection located in the same city (Toronto). A network route trace from this client location to the WAX server takes 15 hops, while a trace going the opposite direction takes 12 hops.

### 5.2.2.1 Video Stream

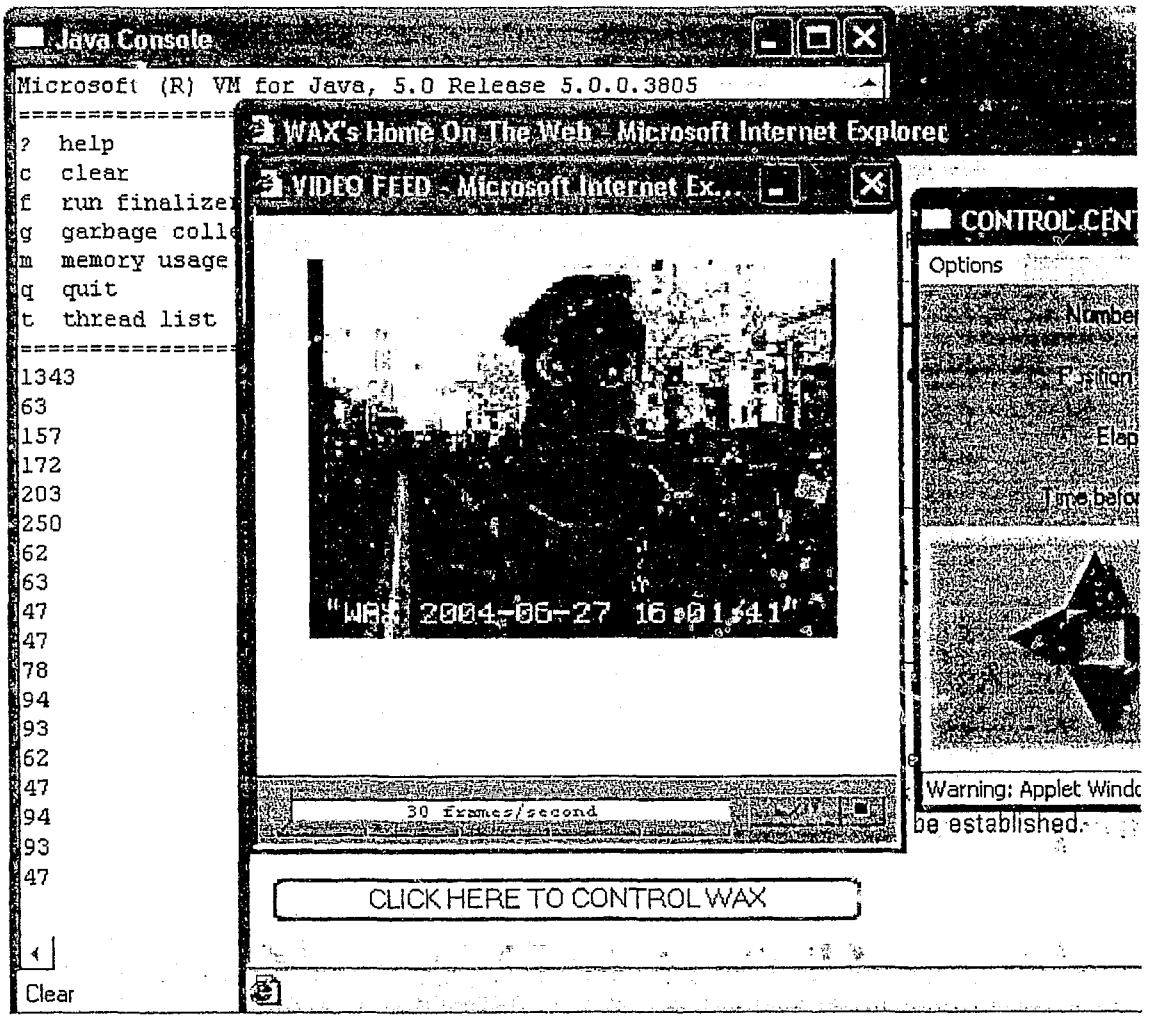


Figure 5.34 WAX video streaming on the Internet

When the WAX client is run from the Internet, the average delay for receiving the video frame is increased from 0-10 ms for LAN to 47-94 ms. Also the chance of network jitter is increased. As shown in Figure 5.23, there were 4 occasions of late video compared to just one during the test using campus network connection.

### 5.2.2.2 Audio Stream – 100 ms

This test is performed on the same machine booted to Linux. For consistency, Netscape 4.8 is used.

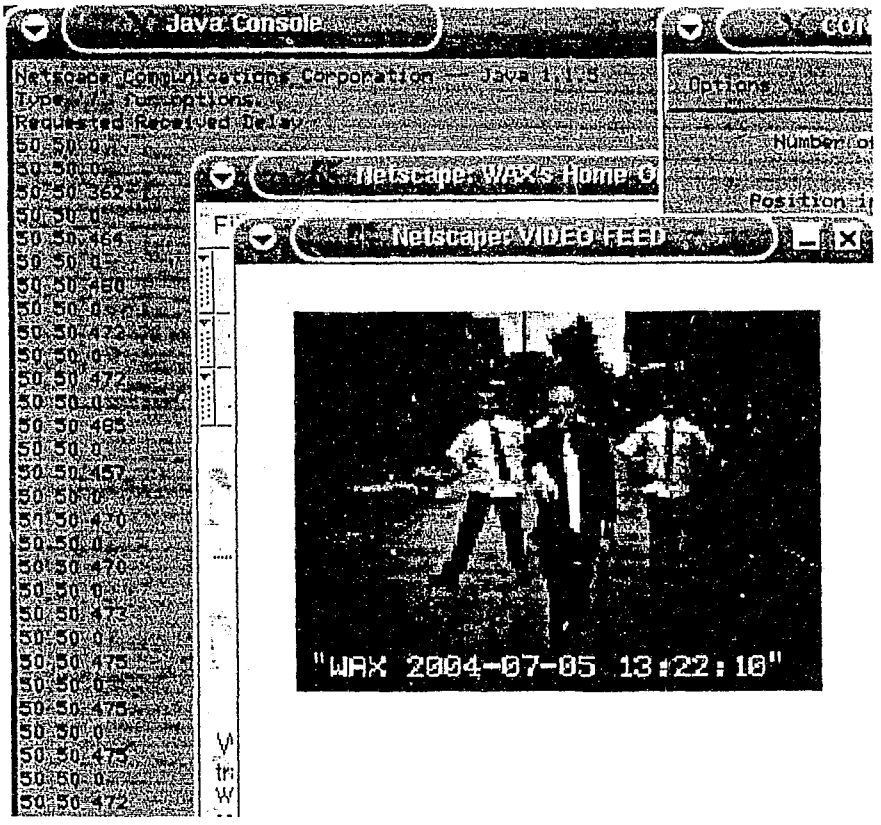


Figure 5.35 WAX audio streaming on the Internet with 100 ms audio MU

With 100 ms audio MU, the WAX client simply can not maintain a smooth audio playback. There is a 400 ms pause for every audio MU.

5.2.2.3 Audio Stream – 200 ms


Figure 5.36 WAX audio streaming on the Internet with 200 ms audio MU

With 200 ms audio MU, the WAX client is able to achieve a much smoother audio playback. There are about 2 pauses for every second of audio.

5.2.2.4 Audio Stream – 300 ms

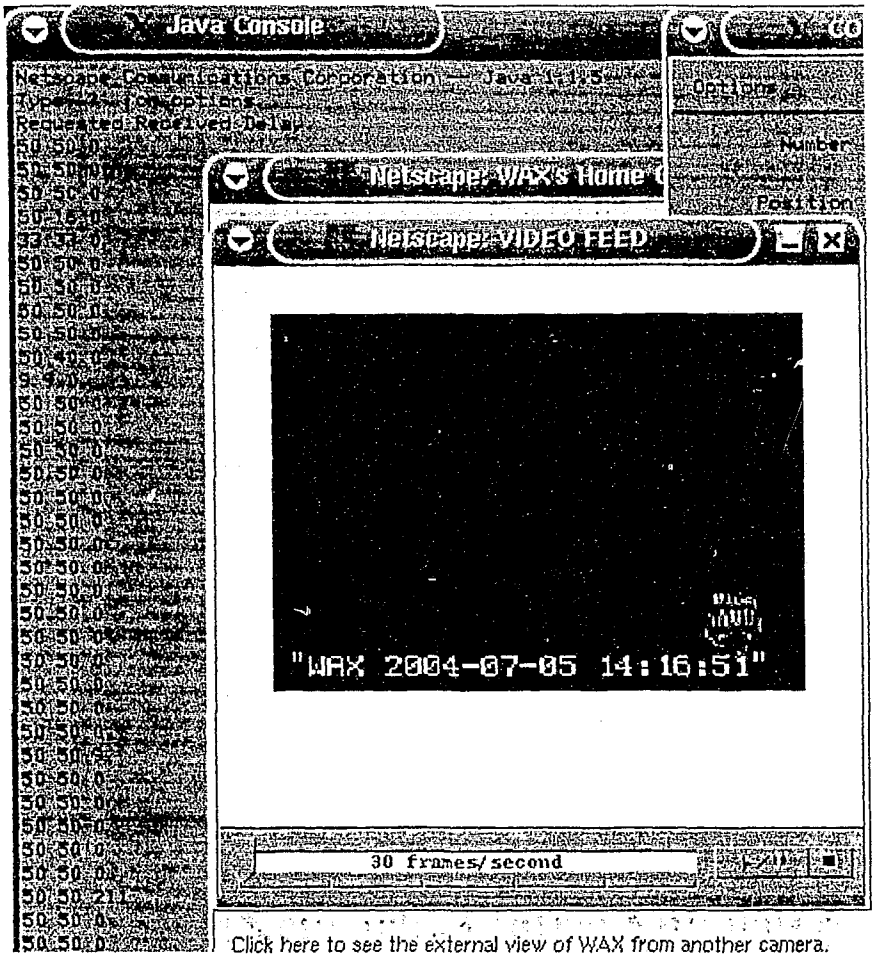


Figure 5.37 WAX audio streaming on the Internet with 300 ms audio MU

With 300 ms audio MU, the pauses in the audio have reduced to once every two seconds.

5.2.2.5 Audio Stream – 400 ms

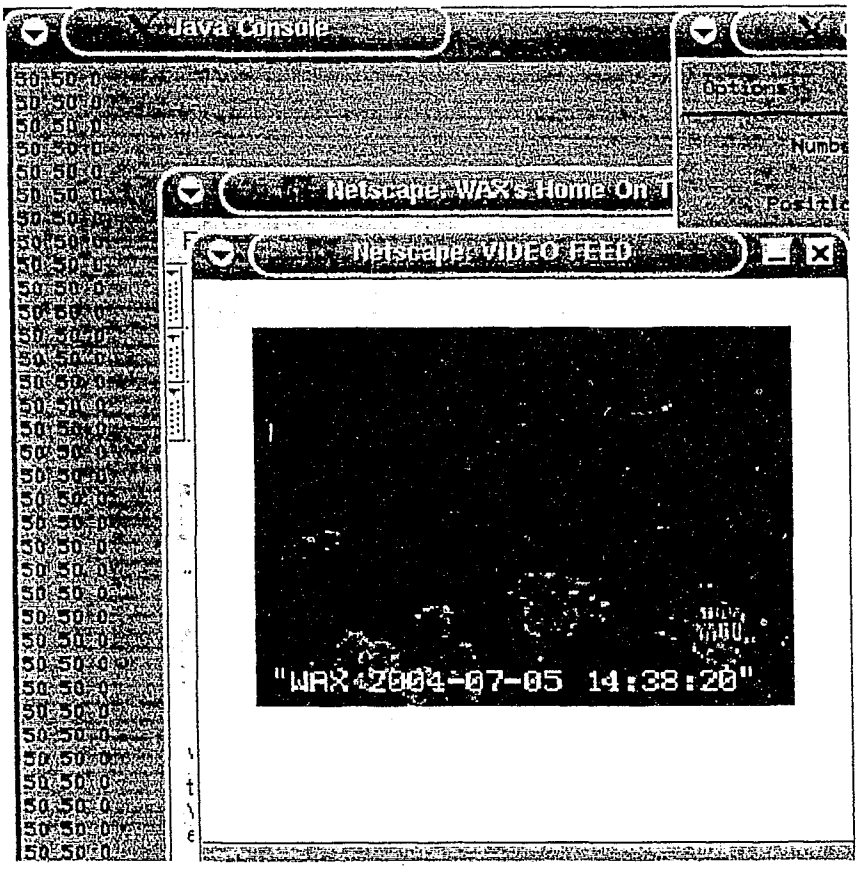


Figure 5.38 WAX audio streaming on the Internet with 400 ms audio MU

As illustrated in Figure 5.27, there is no pause in the audio playback with 400 ms audio MU is used on Linux Netscape 4.8.

To ensure the result is consistent on other operating systems. This test is repeated on the same machine booted to Windows XP with Netscape 4.8.

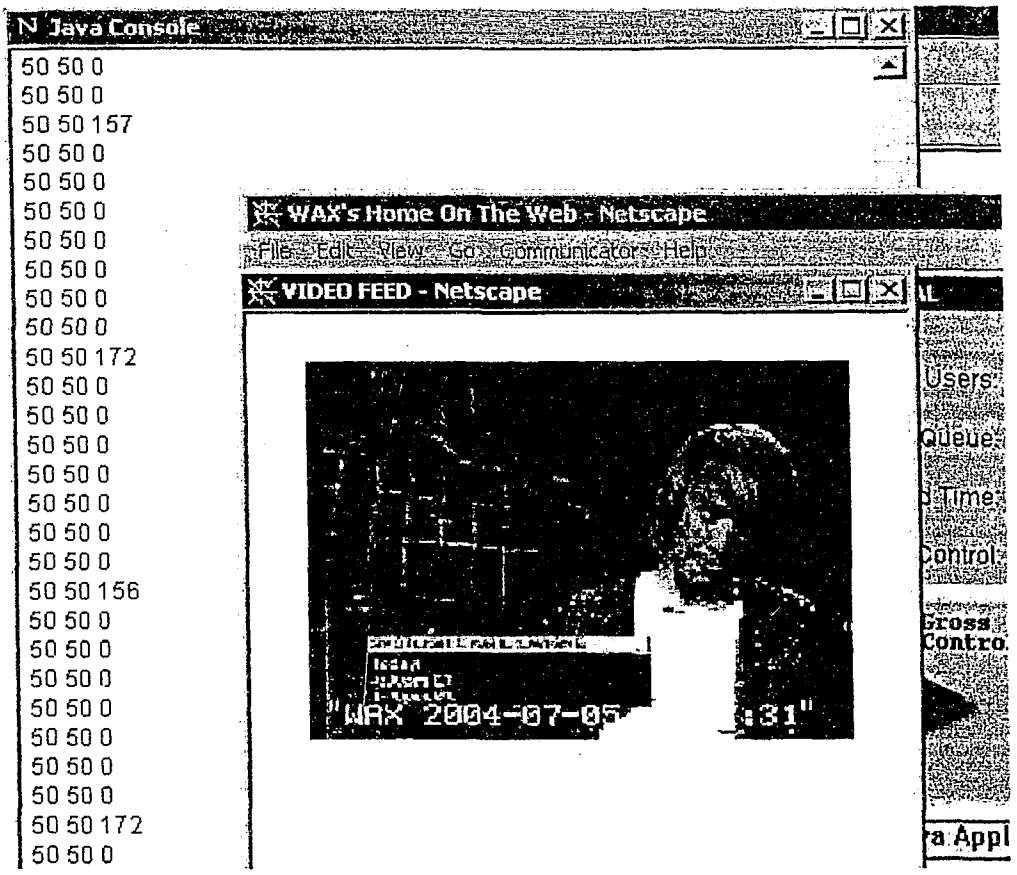


Figure 5.39 WAX audio streaming on Netscape for Windows with 400 ms audio MU

Although Netscape is able to achieve perfectly smooth audio playback on Linux, it cannot obtain the same performance on Windows. When running on Windows, there are about two pauses for every second of audio, each pause lasts between 150-170 ms.

The test is repeated on the same machine and the Windows OS with Internet Explorer 6.

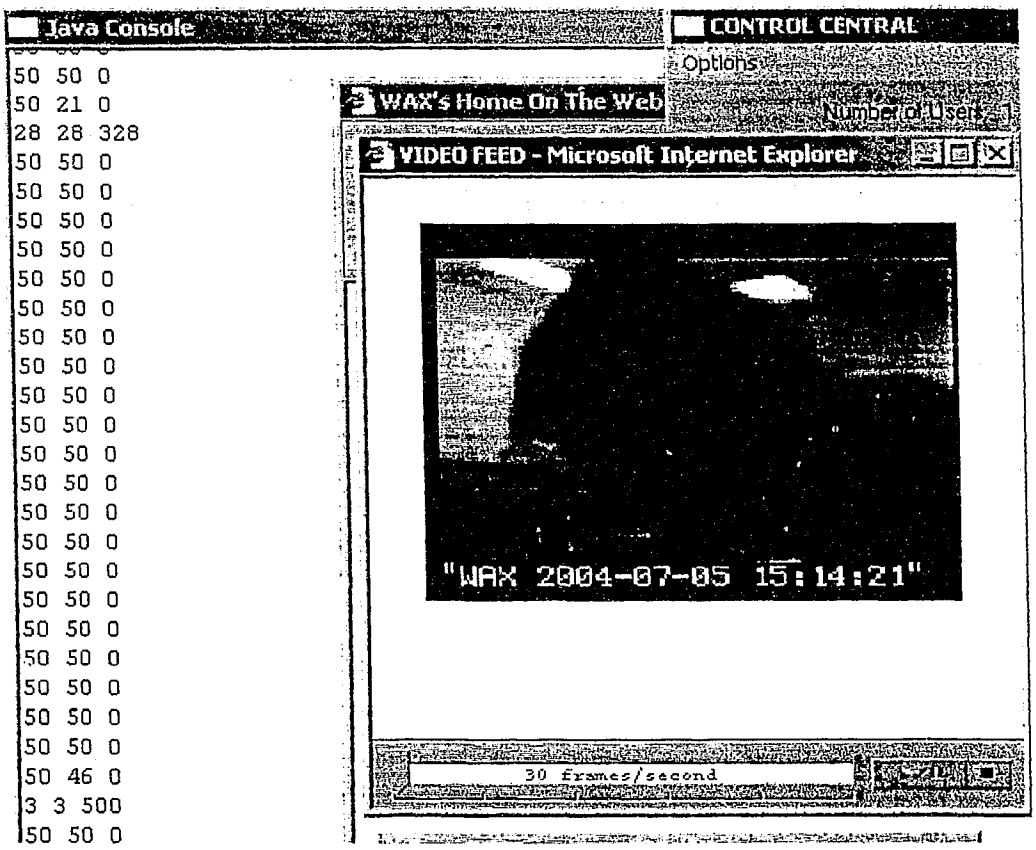


Figure 5.40 WAX audio streaming on Internet Explorer 6 with 400 ms audio MU

When the WAX client is running on Internet Explorer, although there is only one pause for every second, the pause last longer than on Netscape 4.8 and is between 300-500 ms.

### 5.2.2.6 Audio Stream – 500 ms

The test is repeated with 500 ms audio MU.

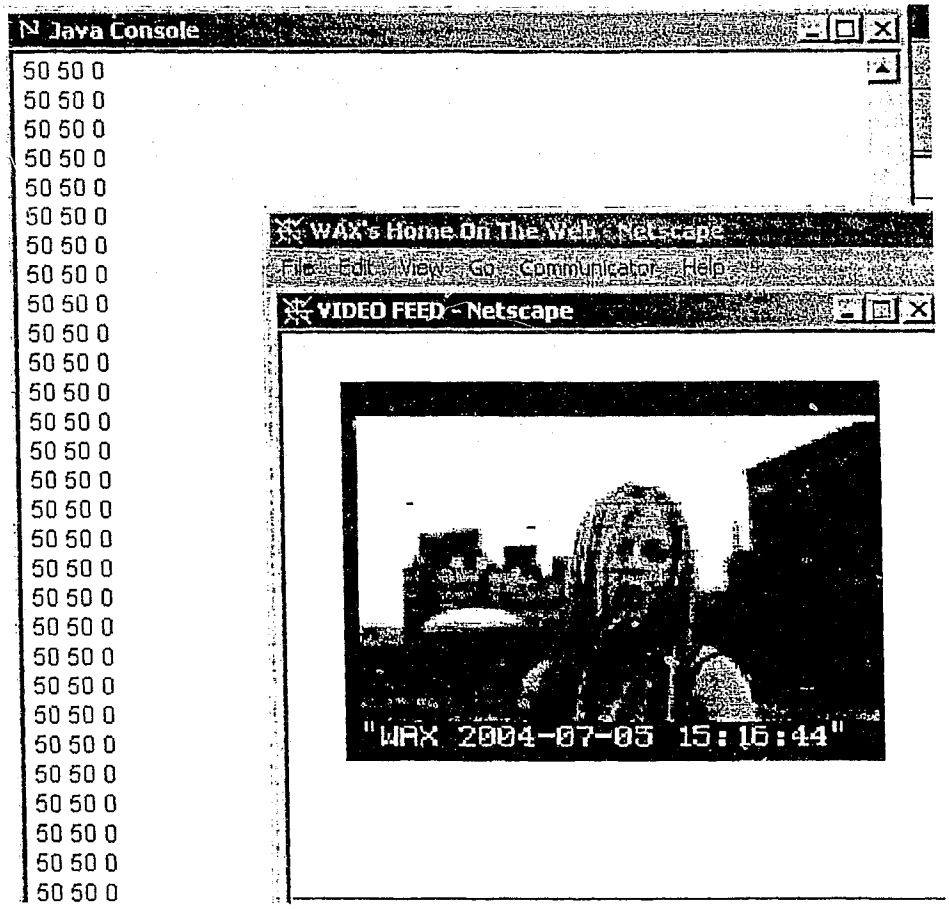


Figure 5.41 WAX audio streaming on Netscape 4.8 for Windows with 500 ms audio MU

With 500 ms, Netscape 4.8 is able to maintain a continuous audio playback without pauses.

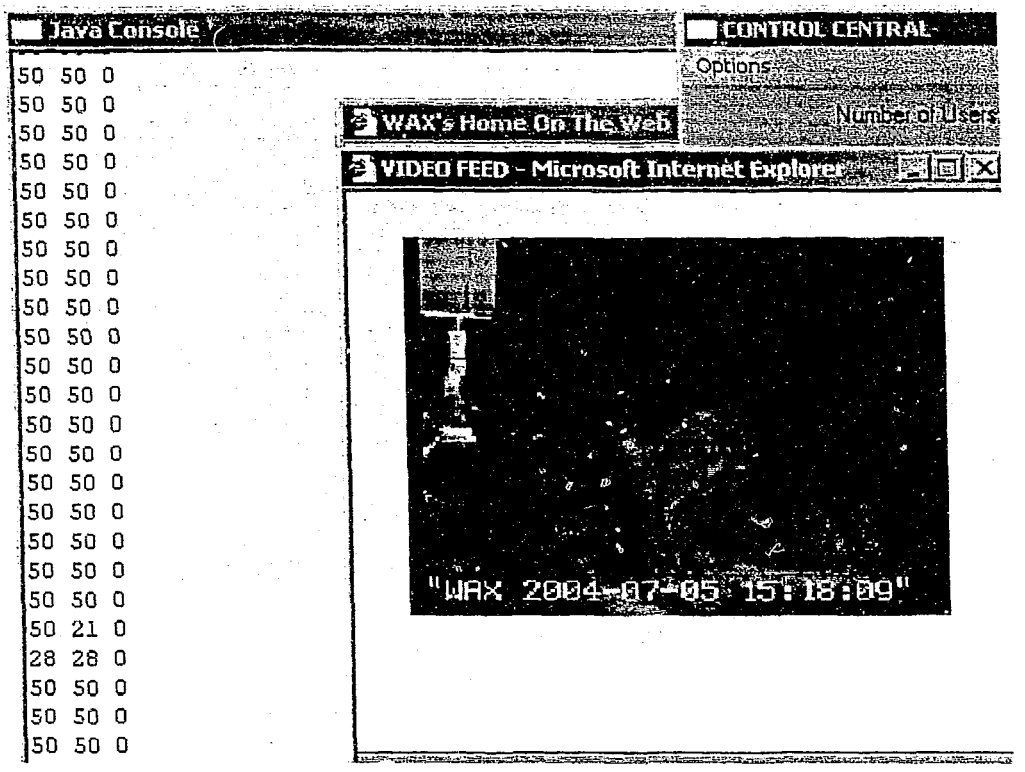


Figure 5.42 WAX audio streaming on Internet Explorer 6 with 500 ms audio MU

When 500 ms audio is used, Microsoft Internet Explorer 6 is also able to maintain a smooth audio playback.

The test is repeated with Netscape 7.1 with bundled Sun Java.

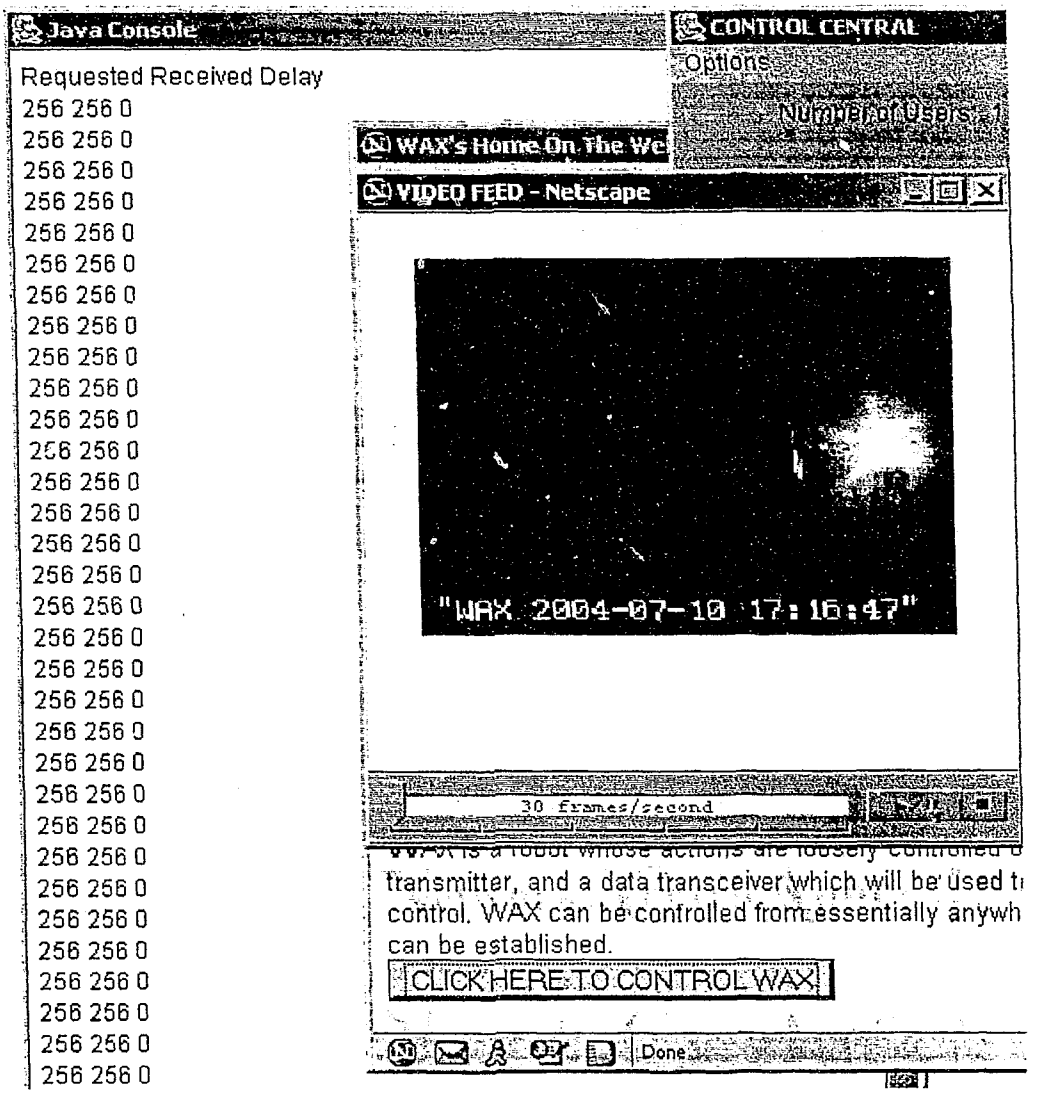


Figure 5.43 WAX audio streaming on Netscape 7.1 with 500 ms audio MU

As illustrated in Figure 5.32, the new implementation of Sun's Java 2's audio engine uses 256 ms for the audio buffer. WAX client is still able to playback the audio without pauses.

From the experiments over the Internet, the time delay of the audio stream with different MU size can be plotted against the MU sequence.

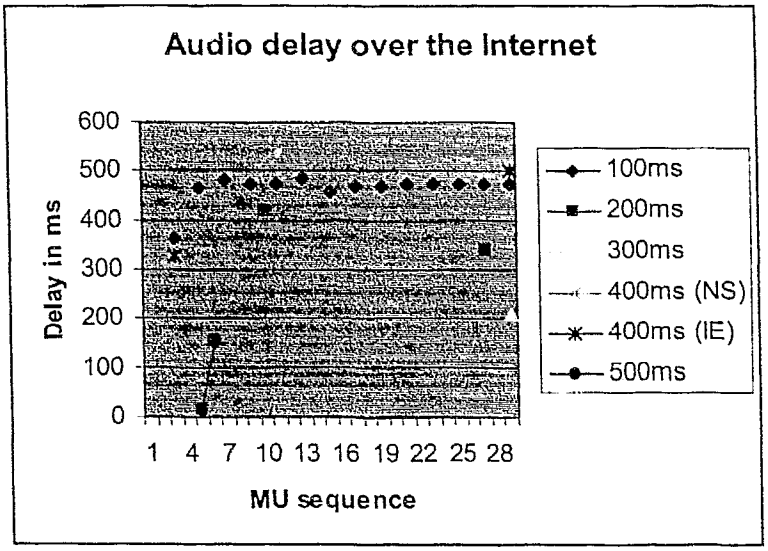


Figure 5.44 Audio delay over the Internet

As illustrated in Figure 5.44, with 100 ms audio MU, there are consistent and frequent pauses in the audio stream. However, with 200 ms audio MU, the occurrences of pauses reduced from 14 to 4, but the duration of the delay can still be as high as with 100 ms. For 400 ms audio MU, audio streaming on Internet Explorer produced more pauses than with the same client is ran on Netscape 4. However, the pauses in Internet Explorer last shorter than the ones in Netscape 4.

## 5.3 Inter-Stream Synchronization

Since 500 ms of audio MU can provide smooth audio playback for both LAN and the Internet, it is selected as the final size for the audio MU, it is used for the inter-stream synchronization test for both LAN and the Internet.

The inter-stream synchronization performance of WAX is more difficult to measure than intra-stream synchronization due to the following:

- The majority of the inter-stream synchronization is done at the server side. The WAX client has no knowledge of timing of the streaming data except to present them in order. However, the measurement cannot be done at the server side since network delay should be taking into account, and the inter-stream synchronization performance of the playback should be done at the client side.
- WAX uses adaptive source skipping. Therefore using timestamps of each video frame and audio MU is not an accurate measurement.
- Inter-stream synchronization measurement is more content-based than time-based. In other words, human speech should match with the video image, rather than according to proposed timestamp data.

Therefore for this test, only human actors are used to check the inter-stream synchronization performance of WAX. This is a subjective approach since humans cannot measure the errors as accurately as the machines do. However, humans can detect asynchrony in the content of the video and audio which machines are unable to perform.

## 5.3.1 Local Area Network

The first test is to run the WAX client on the server machine.

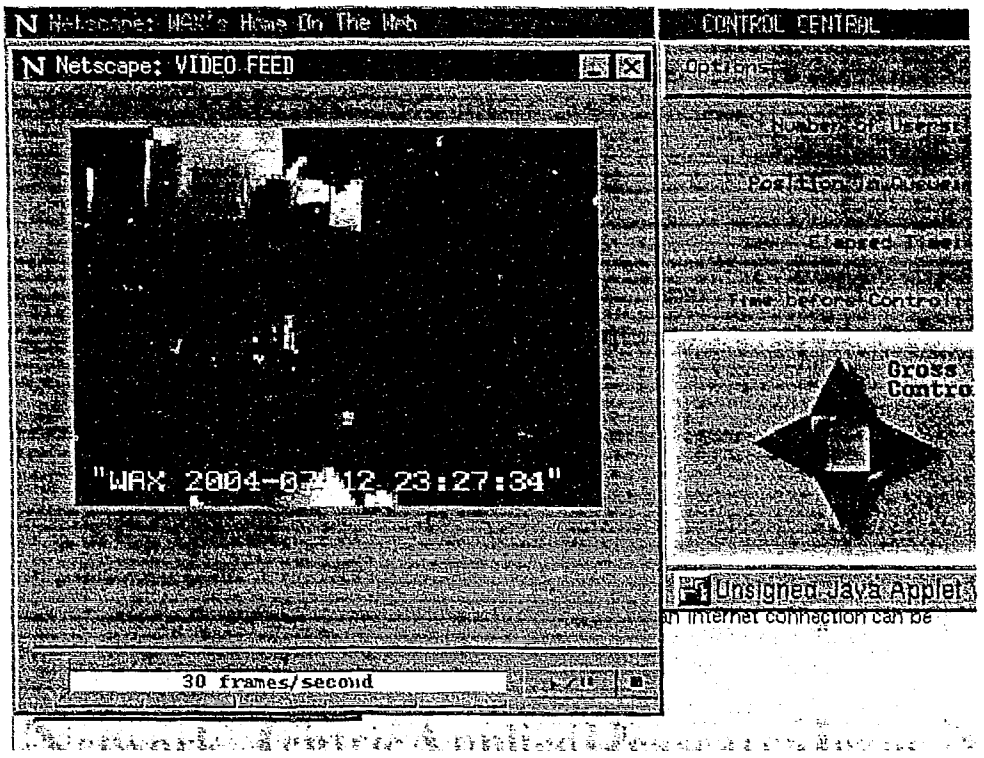


Figure 5.45 WAX inter-stream synchronization test on the server machine

On the server machine, WAX is able to maintain near-perfect synchronization. This is mainly due to the fact that there are no dropping video frames or audio MUs.

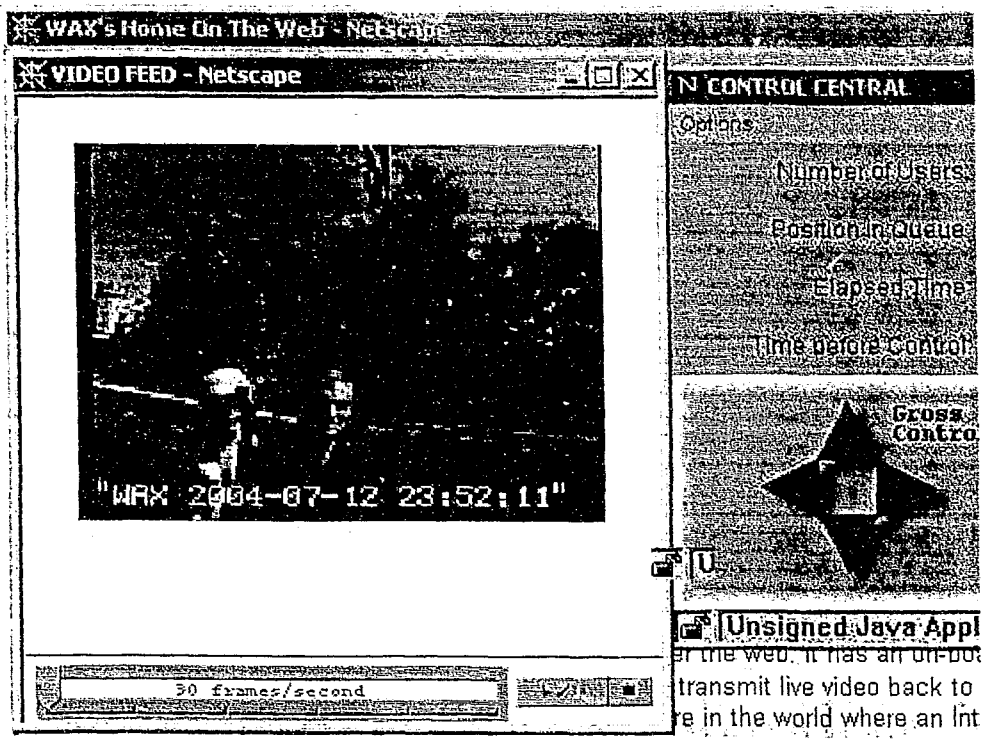


Figure 5.46 WAX inter-stream synchronization test on a LAN machine

When on a LAN machine, the WAX client can achieve the same degree of inter-stream synchronization as on the server machine.

Since Sun's new implementation of Java uses a different buffer size for audio. The test is repeated with Netscape 7 on the same LAN machine.

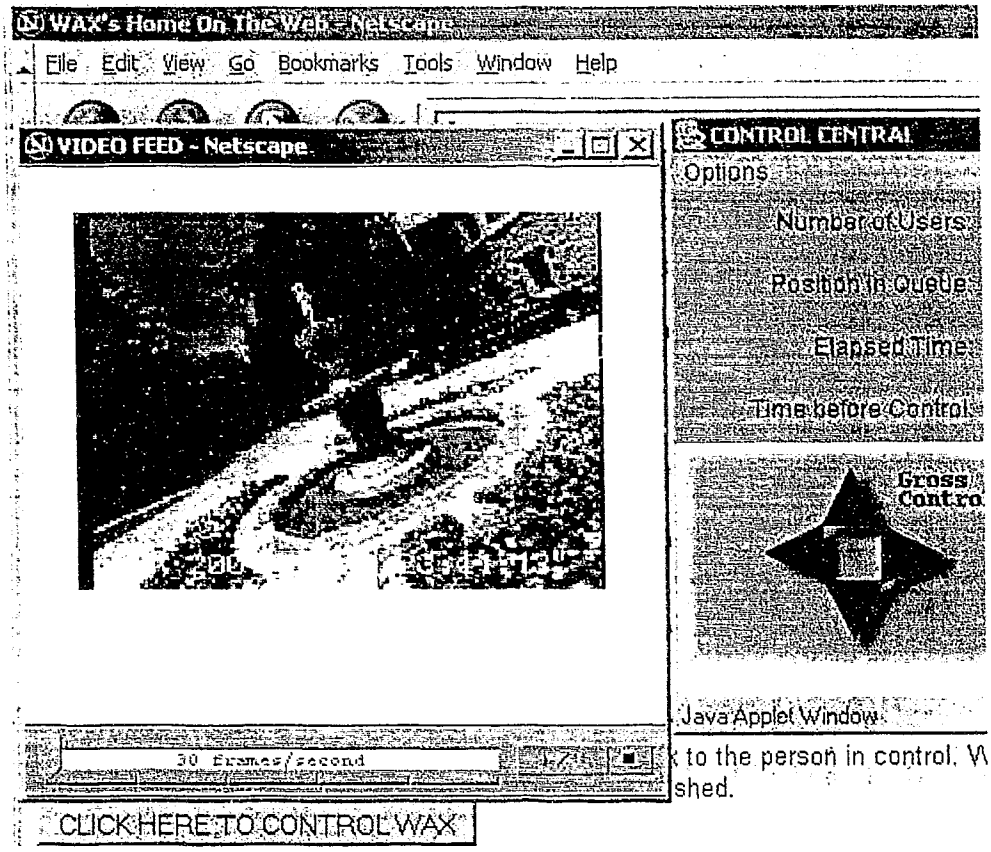


Figure 5.47 WAX inter-stream synchronization test on Netscape 7

As expected, the audio is delayed by approximately 200 ms from the video. It is because the new Java API needs 256 ms of audio buffer instead of 50 ms on the old Java 1.1.

WAX server can be modified to detect the Java version of the client easily, since WAX's Java client provides a full HTTP header for the WAX server. The header includes information such as operating system, browser type and Java version. The WAX server can then use this information to adjust the video roll-back buffering accordingly. This can also be done at the client side, which buffers images from WAX sever if the Java version is later than 1.1. However, it is simpler to implement this at server side.

The test is repeated on the same campus machine.

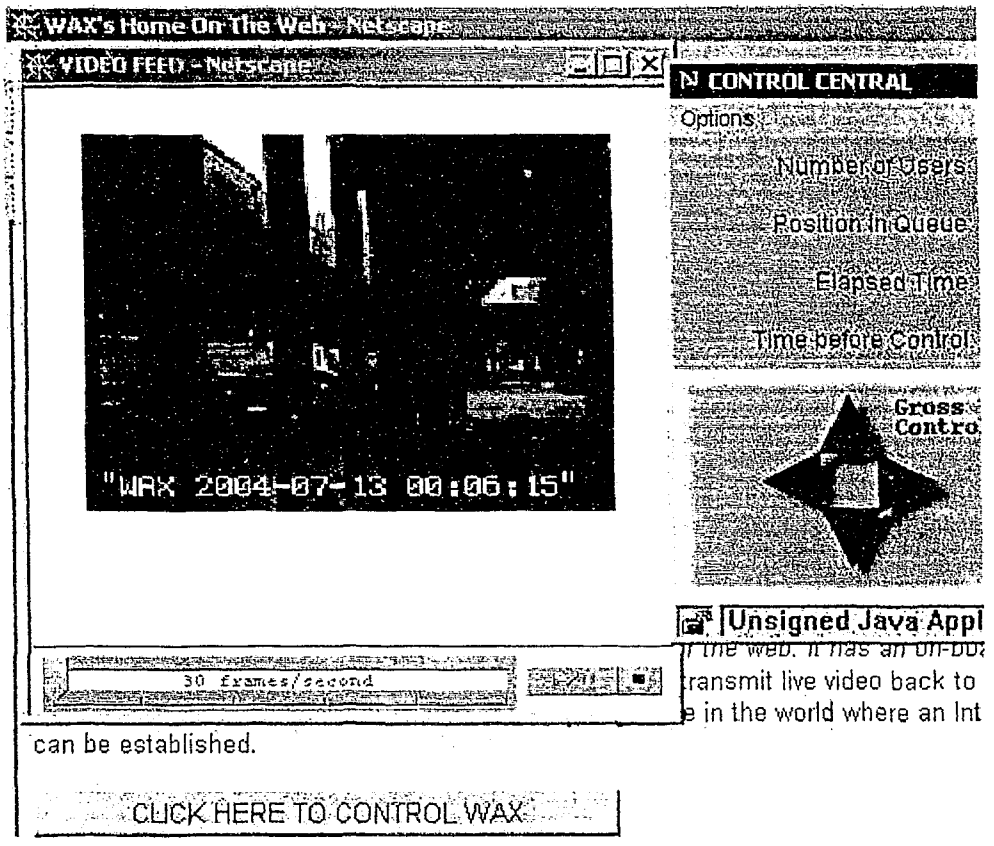


Figure 5.48 WAX inter-stream synchronization test on a campus machine

When WAX client is running on the campus machine, video pauses occur 0-1 times every second while the audio is continuous. Besides the pause in the video, the audio is still in sync with the video.

### 5.3.2 The Internet

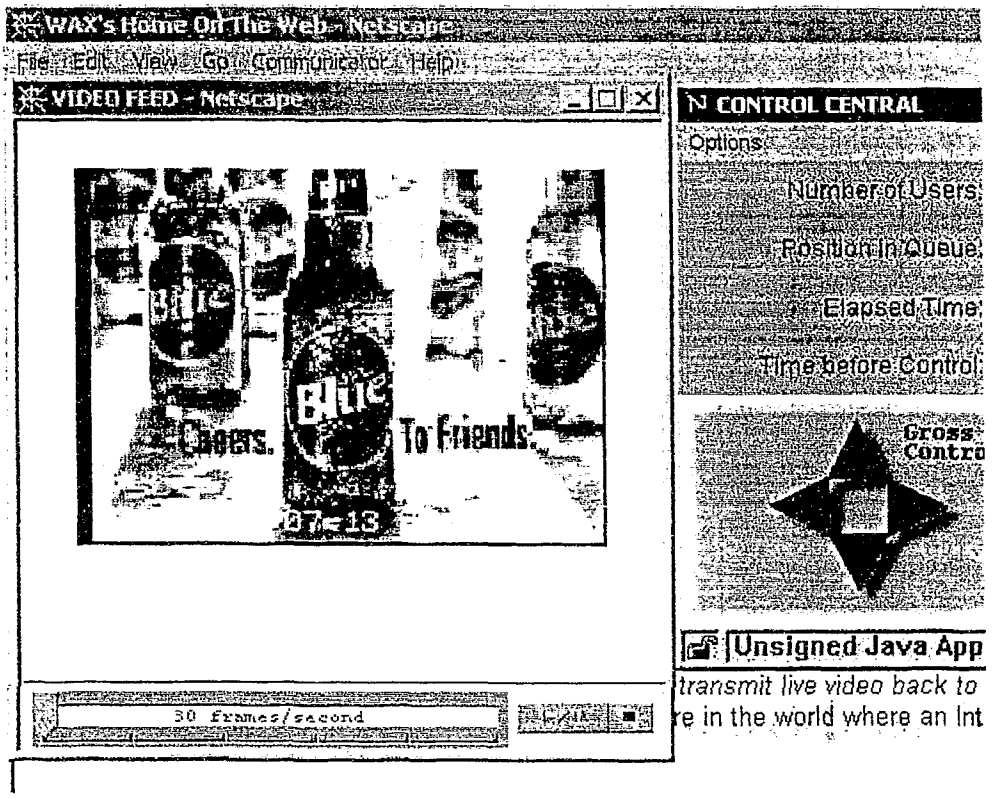


Figure 5.49 WAX inter-stream synchronization test on the Internet

The same 5 Mbps Internet connection is used for the Internet test. With the Internet connection, there are between 0-2 pauses per second in the video. This creates some delay in the video for the audio playback. However, this delay doesn't exceed 100 ms.

## 5.4 Bandwidth Stress Test

Since 5 Mbps connection is already tested with WAX, the bandwidth stress tests use connections lower than 5 Mbps. The three connections used for this test are:

- 2 Mbps Wireless/ADSL/Cable Modem Connection – A wireless router is used for this test to throttle the bandwidth offered by the same Roger's 5 Mbps Internet connection

used in previous tests. This 802.11g wireless router allows the user to change the maximum transmission rate for the wireless connection with its web-based configuration page. A 2 Mbps connection is selected for the wireless bandwidth between the Internet router and the laptop running the WAX client. The DSL service provided by Sympatico Internet Service and cable modem service provided by Rogers Communication, both offer 3 Mbps Internet connections for consumers in Canada. Therefore, if the WAX client can achieve a good streaming performance with 2 Mbps connection, it should perform better with 3 Mbps connection.

- 1 Mbps Wireless Connection – The same wireless router is used with a 1 Mbps bandwidth cap.
- 56 Kbps Dial-up connection - A dial-up modem with a separate dial-up ISP is used to test the dial-up bandwidth.

### 5.4.1 2 Mbps Wireless/ADSL/Cable Modem Connection

For monitoring the bandwidth, a program called "DU Meter" [117] is used to monitor the download and upload bandwidth of the computer. The first test checks for video streaming, and the second test is for audio streaming. The client is running on Windows XP operating system.

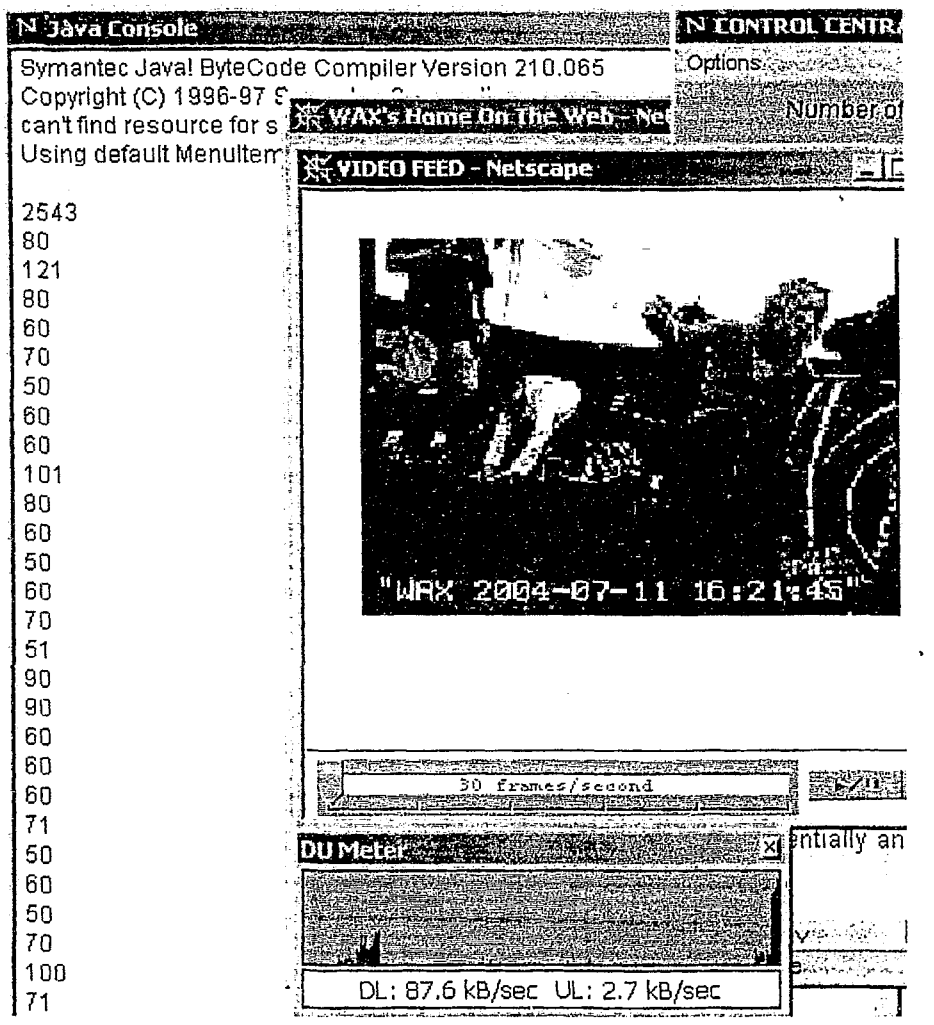


Figure 5.50 WAX video streaming on 2 Mbps connection

According to DU Meter, the total bandwidth for both video and audio streams fluctuate around 60-90 kilobytes per second. This value is close to the network bandwidth required for WAX to achieve full frame rate and smooth audio, which is 88 kilobytes per second (704 kilobits/s). The fluctuation is caused by the various sizes of JPEG images due to image complexity.

As illustrated in Figure 5.33, the WAX client is able to achieve the same quality of video streaming as if it is 5 Mbps. The late frames do not exceed 4 frames every second.

These late frames are mainly caused by network jitter and cannot be avoided.

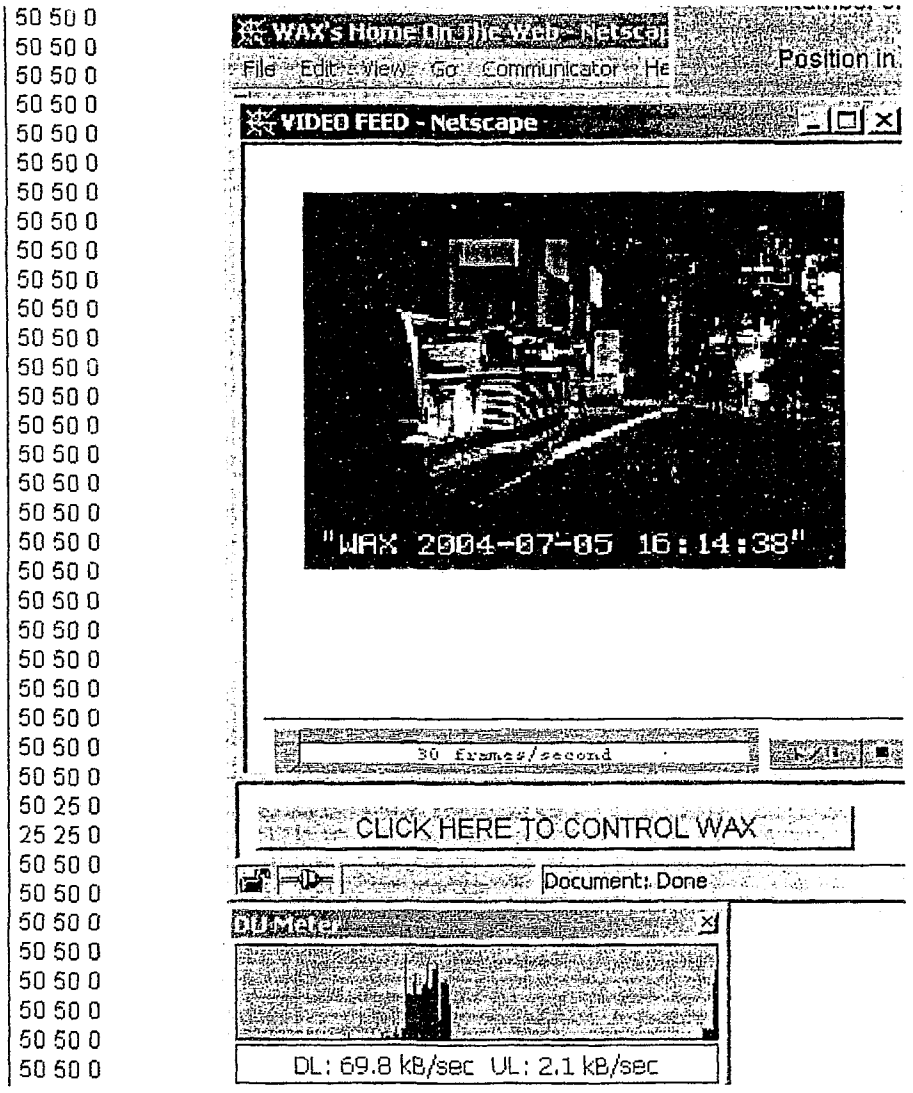


Figure 5.51 WAX audio streaming on 2 Mbps connection

As shown in Figure 5.34, the WAX client is able to maintain smooth audio with 2 Mbps wireless Internet connection.

## 5.4.2 1 Mbps Wireless Connection

The test is repeated with 1 Mbps wireless Internet connection.

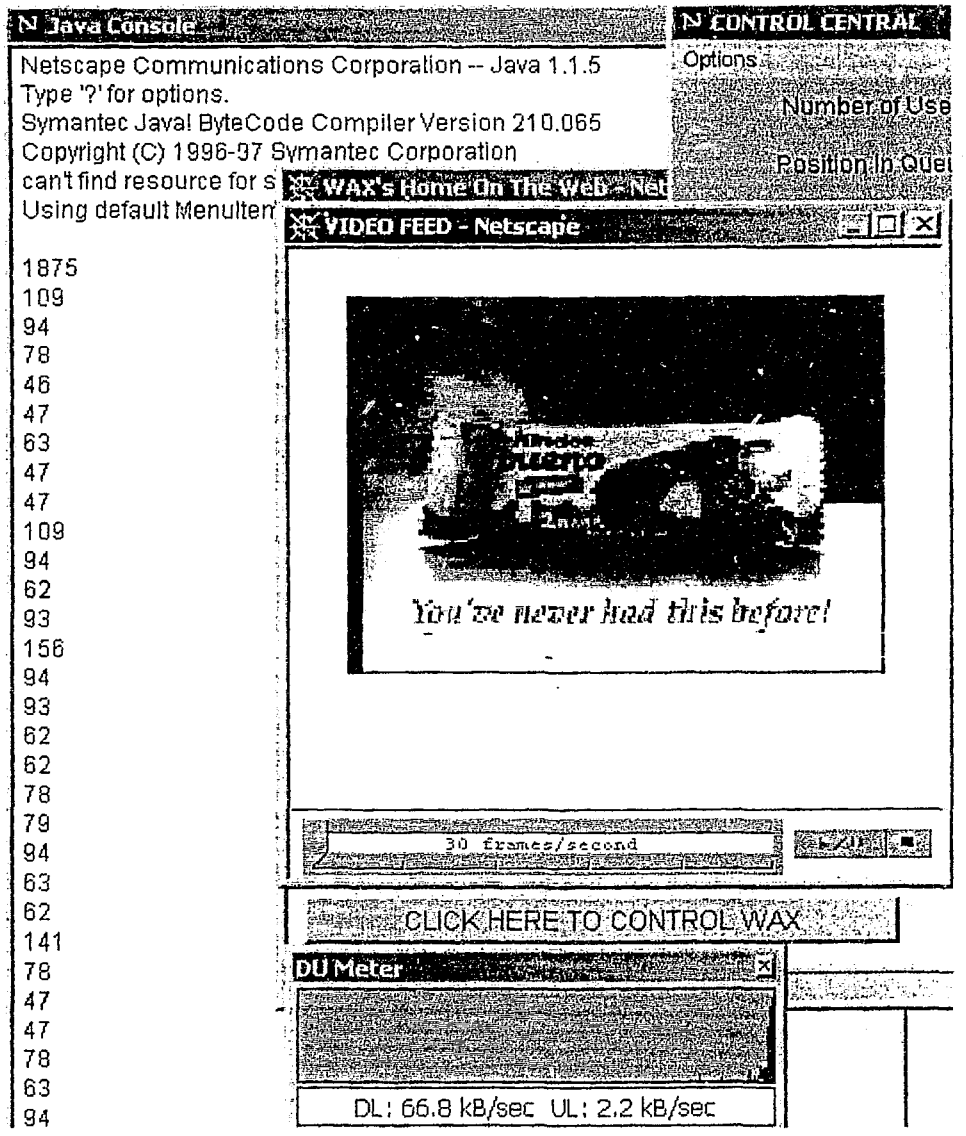


Figure 5.52 WAX video streaming on 1 Mbps connection

With 1 Mbps connection, the WAX client is still able to achieve the same streaming performance as with 2 Mbps connection, since the bandwidth is still higher than WAX's 704 kilobits/s required bandwidth.

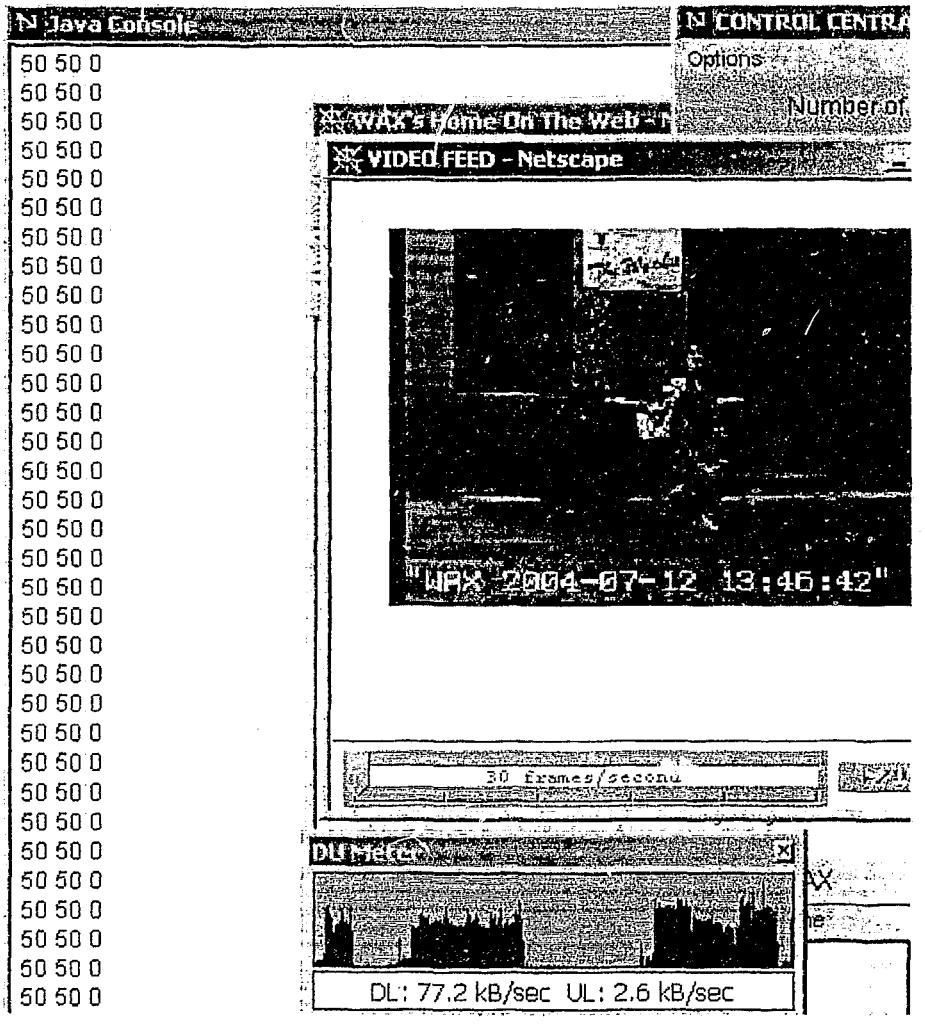


Figure 5.53 WAX audio streaming on 1 Mbps connection

As shown in Figure 5.36, WAX client can still maintain a smooth audio playback with 1 Mbps connection.

## 5.4.3 Dial-Up Connection

The dial-up connection is from a 56K dial-up account with 295.ca.

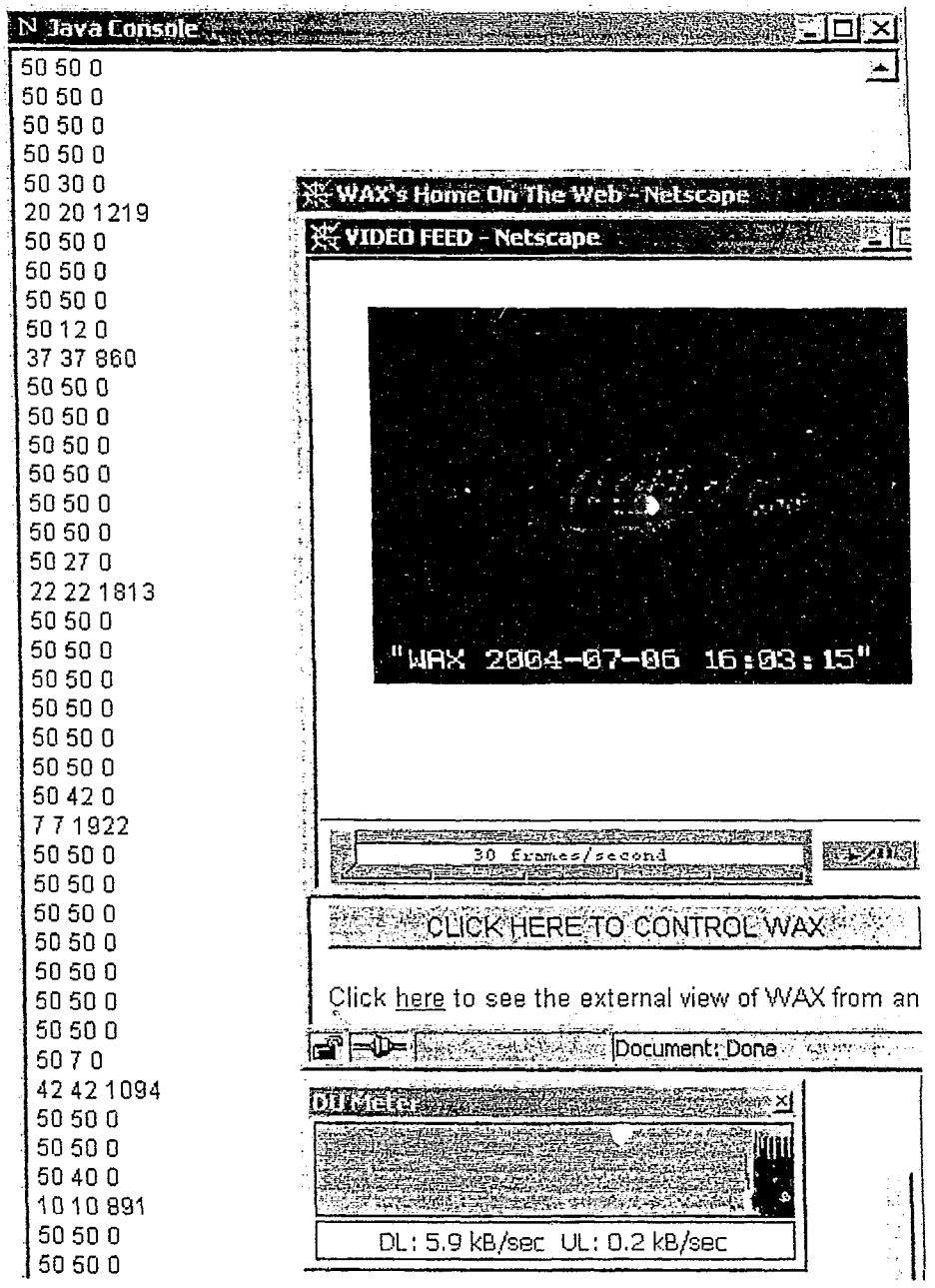


Figure 5.54 WAX audio streaming on dial-up connection

With the dial-up connection, there are pauses in WAX's audio playback. The pauses occur about every 400 ms. The pauses can be as long as 2 seconds.

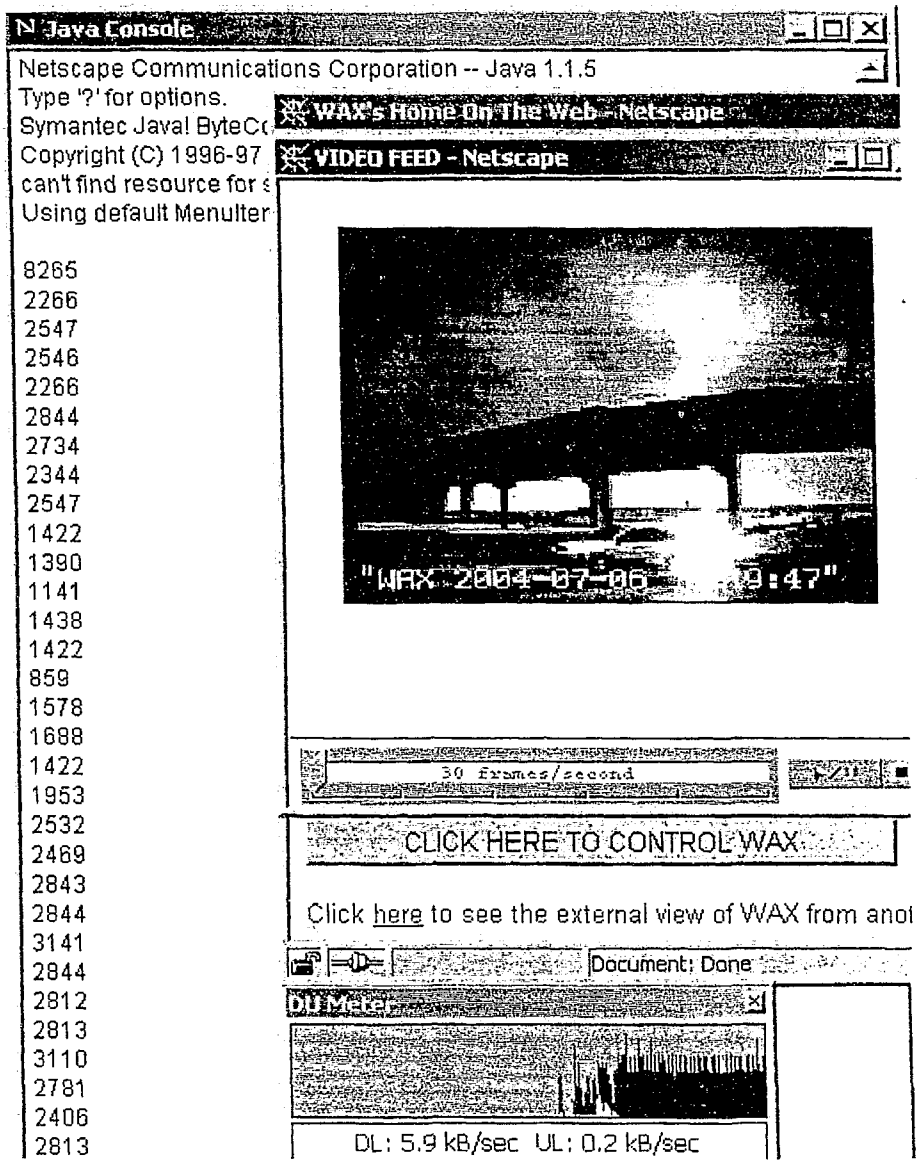


Figure 5.55 WAX video streaming on dial-up connection

On average the WAX client is only able to obtain a frame every 1.5 to 3 seconds with a dial-up connection. Most of the bandwidth is used to provide continuous audio.

Although the video is delayed, the video images provided by the WAX server are quite recent due to adaptive source skipping. This can be verified by the time imprinted on the video image. Adaptive source skipping also prevents the video stream from drifting. Drifting

is when video streaming is getting further behind the audio stream as time elapses. Drifting occurs when excess late streaming data is not discarded but rather accumulated in the buffer.

## 5.5 Portability and Compatibility

This test checks the portability of the WAX client on different hardware and OS platforms, and its compatibility with different Java-enabled web browsers. This test is independent of network bandwidth. Therefore it is carried out where test facilities (hardware and OS) is available.

There are three types of Java for web browsers:

- Sun's official Java VM which adds Java capability to web browsers.
- Microsoft's own implementation of Java VM which can be used in Internet Explorer 4.x, 5.x, and 6.x.
- Sun's old Java 1.1 which is included as a plug-in for Netscape 4.x

Netscape 6 and later, Opera, and other existing non-Microsoft browsers all use Sun's Java for Java contents, since Java is the proprietary technology from Sun. Therefore, if the WAX client works under Sun's Java, then it works on all web browsers which uses Sun's Java, including Netscape, Mozilla, and Opera.

For Internet Explorer 4 and up, the user can choose either to use Microsoft's own Java virtual machine, or Sun's Java.

Netscape 4.x includes Sun's Java 1.1 which is bundled as a plug-in with the browser under their license agreement with Sun. Netscape 4.x is still used in many places running low-end systems.

## 5.5.1 Sun's Java VM

The WAX Java client was tested with Sun's Java bundled with Netscape 7.1 on Windows XP.

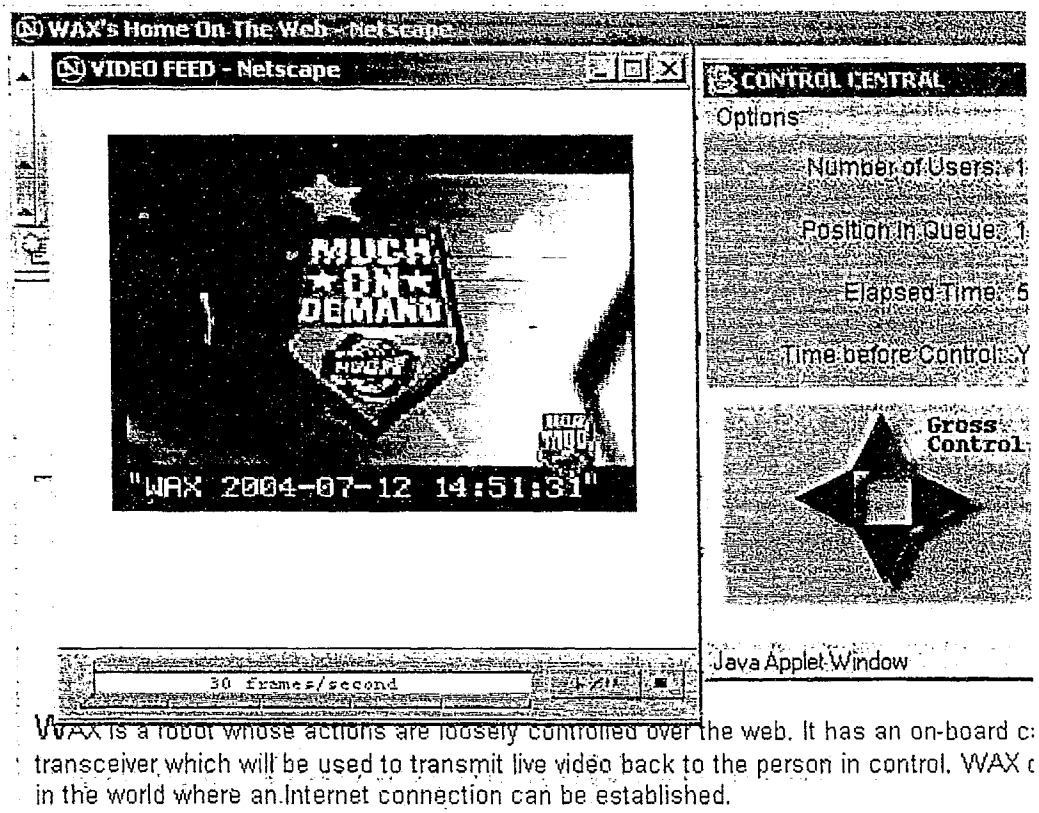


Figure 5.56 WAX client on Netscape 7.1 with bundled Sun's Java

Both video and audio are working. From previous tests, it is shown that the new implementation of Sun's Java uses 256 ms of sound buffer, instead of 50 ms in previous implementations.

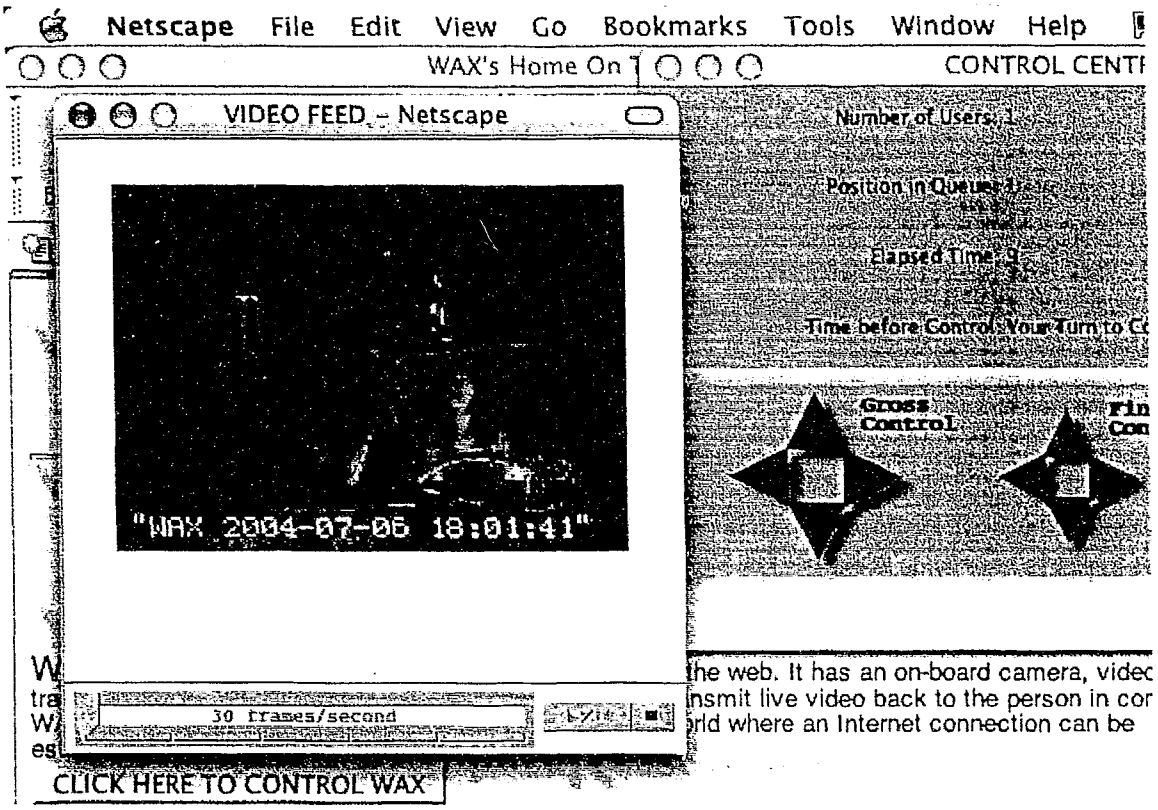


Figure 5.57 WAX client on Netscape 7.1 under Mac OS X

The WAX client also runs on Netscape 7.1 for Mac OS X 10.3. Note the missing WAX menu in the control center panel. For some reason the Java applet's menu is not displayed on Mac OS X.

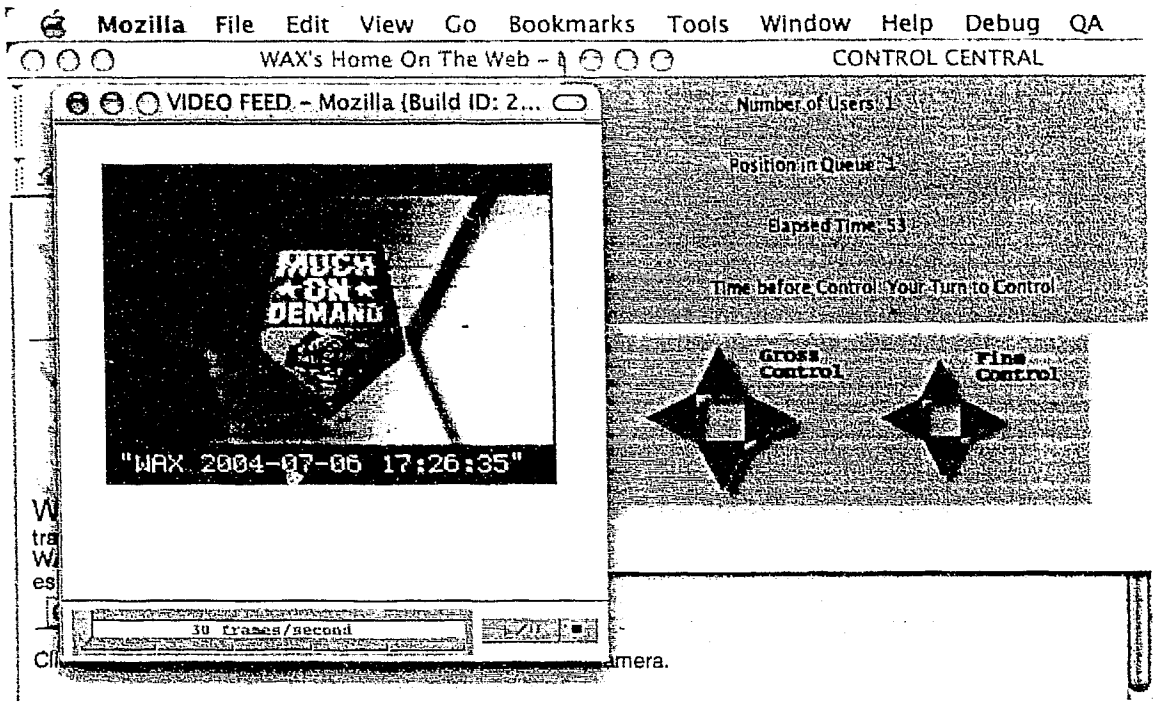


Figure 5.58 WAX client on Mozilla under Mac OS X

As expected, Sun's Java on Mozilla behaves the same as on Netscape 7.

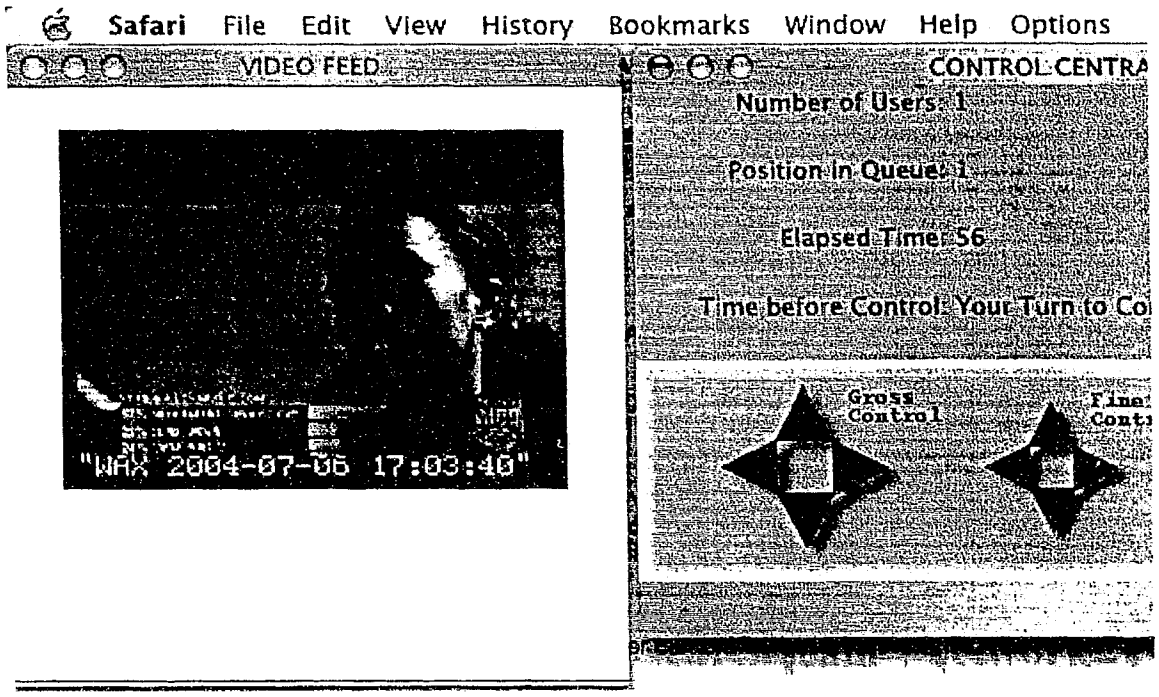


Figure 5.59 WAX client on Safari under Mac OS X

WAX's video and audio streaming also works on Safari for Mac OS X.

## 5.5.2 Microsoft's Java VM

Microsoft's Java VM works under Internet Explorer 4/5/6 and any web browsers which use the Internet Explorer engine, such as SlimBrowser and SmartExplorer.

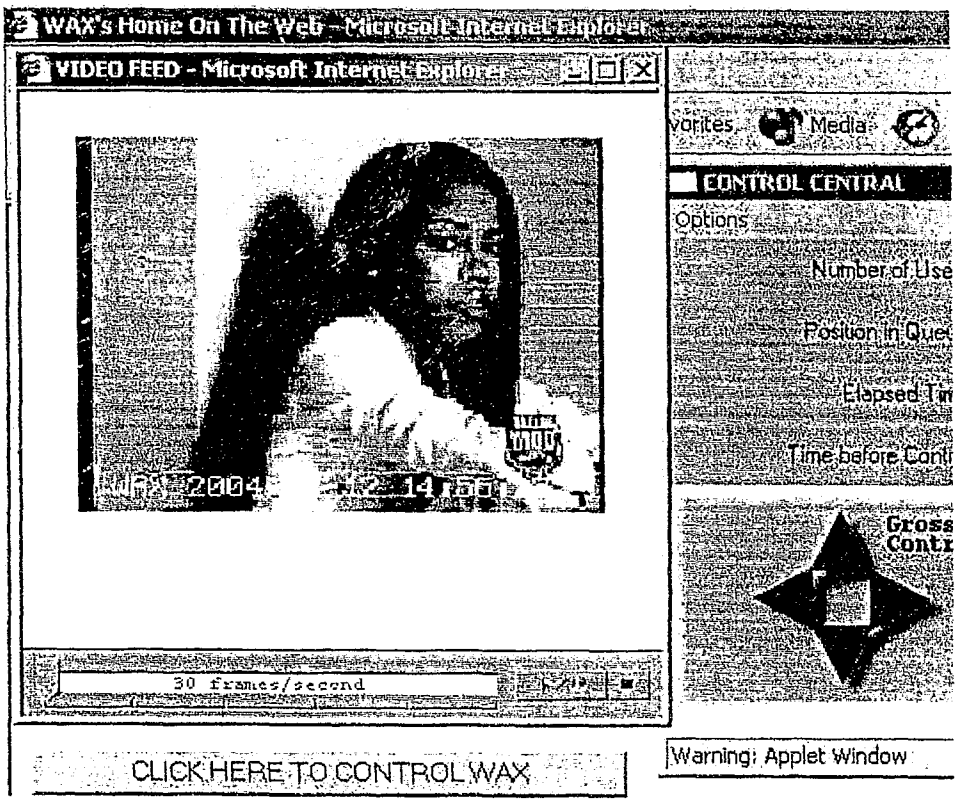


Figure 5.60 WAX client on Internet Explorer

WAX Java client has no problem running under Microsoft's Java VM.

## 5.5.3 Netscape 4.x with Sun's Java 1.1.5

Many of the previous tests use Netscape 4.8, since this browser demonstrates WAX's compatibility with old Java plug-ins. The test is shown here for completeness.

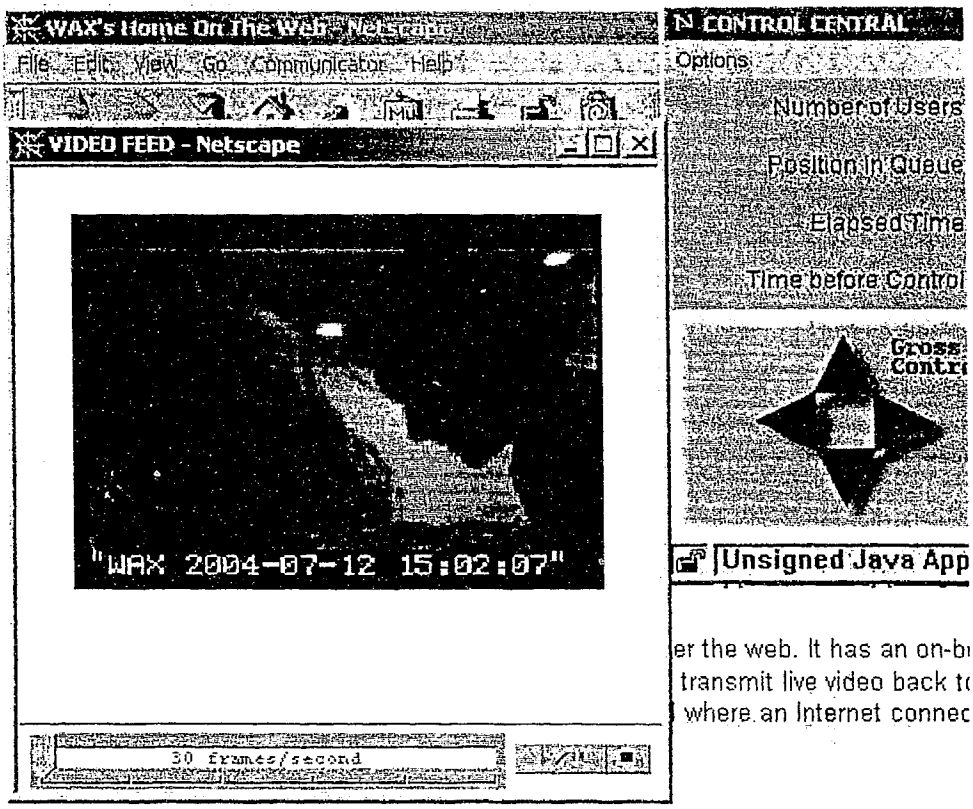


Figure 5.61 WAX client on Netscape 4.8

# 6. Conclusions & Future Work

## 6.1 Conclusions

From this work we have achieved certain significant results and it is possible to draw various conclusions. First of all, for intra-stream synchronization, the following empirical data were obtained:

- Video intra-stream synchronization can be maintained with 10 fps on a LAN as well as the Internet. 10 fps is the maximum frame rate for regular consumer webcams.
- With a LAN, the end-to-end delay of video MU is under 10 ms. While on the Internet, the average delay is between 40-90 ms. These are under the 100 ms frame interval of the 10 fps video stream.
- For a LAN, an audio MU length of more than 400 ms can guarantee the continuity of the audio stream.
- For the Internet, using Netscape for Linux, the minimum audio MU length for smooth audio playback is 400 ms.
- For the Internet, if the Windows OS is used, then the minimum audio MU length is 500 ms.

500 ms is selected as the audio MU length for WAX since it guarantees the continuity of the audio under any conditions. However, in reality, this may be a significant problem for control. In essence at least half a second goes by before the synchronized stream is "experienced" by the operators. If the robot is moving in a dangerous environment, this may be too long of a delay.

For inter-stream synchronization, the following can be concluded:

- In a LAN environment, a near-perfect inter-stream synchronization can be maintained with full streaming
- On the Internet, up to 2 pauses per second can occur due to network delay. However, inter-stream synchronization can still be maintained. The temporary misalignment doesn't exceed 100 ms.

The following were also observed during testing:

- Different network efficiency can be achieved by different operating systems running on the same machine. For example, under Linux, Netscape 4.8 only needs 400 ms of audio MU length to maintain a continuous audio. While under Windows, Netscape 4.8 needs 500 audio MU to prevent pause from occurring.
- Sun's Java 2 (1.4.2) uses a larger audio buffer than its previous version. It uses 256 ms of audio buffer instead of 50 ms in Java 1.1. The WAX server has to detect the Java version and change the streaming parameters to compensate for the difference in Java's buffer size.

From the bandwidth stress test, it was shown that:

- With a connection of 1 Mbps or higher, the WAX client is able to achieve full streaming of video and audio
- Dial-up connections don't have the required bandwidth for full streaming of video and audio for WAX. However, the WAX client is able to maintain a relatively smooth audio playback, which lasts 300 ms before a pause occurs.
- With dial-up connections, the WAX client is able to obtain a video frame every 1.5 – 3 seconds. Most of the bandwidth is dedicated to provide continuous smooth audio.

Although the frame rate is low, the video images are recent and drifting does not occur.

The last test demonstrated that the WAX client is portable to many hardware and OS platforms, as well as being compatible with many Java-enabled web browsers. The hardware and OS platforms which WAX can run on include, but are not limit to:

- Windows 9x/2000/XP
- Mac OS X
- Linux/Solaris/*NIX

The Java-enabled web browsers which WAX can run on include, but are not limit to:

- Netscape 4.x
- Netscape 7/Opera/Moziila with Sun's Java
- Internet Explorer (IE) 4/5/6
- IE-based web browsers such as SlimBrowser, SmartExplorer, etc

## 6.2 Summary of Contributions

The WAX system demonstrates successful real-time video and audio synchronization over the Internet, using the multi-stream approach. The WAX teleoperating system employs the fusion of a number of techniques to achieve synchronization. For intra-stream synchronization, the WAX server uses:

- Priority-based transmission
- Static transmission rate adjustment
- Static input rate adjustment

For inter-stream synchronization, the WAX server uses:

- Transmission rate adjustment
- Buffering time adjustment with network delay estimation
- Adaptive source skipping
- Media scaling

While the WAX server is responsible for the synchronization, the WAX client is responsible for the playback of each stream according to its playback rate. This arrangement is used because:

- The WAX server has more information about the streams and has direct control over them.
- It is more efficient to manage synchronization at the server side. For example, the server can reduce the transmission rate if overflow occurs, rather than the client dropping excessive data, which the bandwidth is already wasted.
- The WAX client has less processing overhead, which allows it to run more efficiently and responsively, even on low-end systems.

Although the WAX client does not directly control stream synchronization, it employs a number of programming techniques to manage the playback of the audio stream:

- To make audio streaming possible, the WAX client utilizes Java's undocumented, low level sun.audio.* class package.
- The WAX client uses the streaming buffering technique instead of double buffering technique to achieve a smoother audio playback without causing any addition delay.
- A new streaming class was developed for the WAX client to monitor the performance of the synchronization.

All the techniques that the WAX server and client have employed, except for the

adaptive source skipping, are used for synchronization under normal conditions. The experiment shows that with 1 Mbps connection or higher, the WAX teleoperation system can operate normally.

Under dial-up connections, the WAX system operates in fail-safe mode, which sends video and audio data as fast as possible, and only relies on adaptive source skipping to provide synchronization. The experiment has demonstrated that WAX's adaptive source skipping is able efficiently utilize the available bandwidth, and provide images and audio as quickly as possible. The WAX system also is able to keep the video stream from drifting away from the audio stream.

The WAX synchronization algorithm has demonstrated the flexibility of multi-stream synchronization in comparison to single-stream synchronization, in a number of ways:

- The stream transmission rate can be adapted to each client's connection speed.
- Streaming data can be transmitted as soon as it has been captured. There is no encoding overhead.
- Clients with 1 Mbps or faster connection can receive full video and audio without affecting clients with slow connections. Fast frame rate and continuous audio provide more responsive feedback for reliable and safe operation.
- Clients with slow connections can still receive up-to-date images and audio, with the penalty of lower video frame rate. This is more desirable than receiving outdated full-frame rate video.
- With adaptive source skipping, the slave streams do not drift away from the master stream as time goes by.
- Compatible with existing Java-based video-only teleoperating systems

- JPEG images and AU audio are "web-ready." As long as the client's web browser is Java-enabled, the WAX client can run regardless of the hardware, OS or browser type. No additional third-party multimedia decoder is needed.
- Any stream in the streaming process can be stopped and restarted to compensate for limited bandwidth, or if it is not required for operation.

## 6.3 Future Work

This section discusses some of the potential applications of this research as well as some research topics that were not investigated.

### 6.3.1 NEPWAK

The Network-Enabled Powered Wheelchair Adaptor Kit (NEPWAK) [45] is a mobile system that enables remote assistance in navigation for disabled people or those requiring assistance in powered, differentially steered wheelchairs. The NEPWAK also encounters the same problems related to missing auditory information. Although NEPWAK was developed for a different application, the WAX synchronization model can be applied to the NEPWAK as well. The NEPWAK's system implementation is different from WAX. While WAX uses Linux as the server programming platform and Java for its client, the NEPWAK uses the Windows platform for both server and client. Despite the difference in operating system architecture, some work has been done to enable audio and synchronization on NEPWAK, and it has been successful. With the new implementation, the subject on the wheelchair has the ability to give the operator detailed requests and instructions in completing the tasks allowing a form of cooperative control.

The WAX synchronization model can be applied to any current Internet-based video-only teleoperating system that uses M-JPEG for its video stream. Although WAX synchronization is designed for real-time applications, it is a generic synchronization algorithm which can be used on any real-time or non real-time system.

### 6.3.2 Stereo or Multi-Channel Audio

Stereo audio or multi-channel surround sound will improve the operator's sensory experience of the teleoperated system. With multi-channel sound, the operator is able to identify the direction of the sound source. It could be potentially very useful in applications such as search and rescue to locate survivors.

However, multi-channel audio requires significant amounts of bandwidth. Not only a certain amount of bandwidth must be allocated to each channel but, with a multi-user environment, the bandwidth requirement multiplies each time a new client connects. The robot server can easily become overloaded by relatively few connections. This would also introduce a more complex delay and synchronization model. Therefore the use of multi-channel audio, while potentially useful, must be carefully examined.

### 6.3.3 Bi-Directional Audio Communication

In some cases—NEPWAK or search and rescue robots for example--two-way communication may be desireable. This allows the NEPWAK operator to respond to the requests and warnings from the subject on the wheelchair. For search and rescue, the rescuer can ask the condition of the survivor, provide first-aid instructions and assurance, as well as coordinate and guide the survivor in escaping the hazardous area if possible.

To playback the audio on the robot, the robot needs to be equipped with a transceiver (wired or wireless) which is capable of receiving audio. The robot also needs to have a speaker and power supply to the speaker.

The addition in functionality of the transceiver and the speaker increase the demand in power on the robot. Also the bandwidth requirement for audio stream is much higher than control signals. Each robot command costs at most one byte and an optimized robot server only sends commands to the robot when necessary. The addition of audio streaming may decrease the responsiveness of control commands, and hence can raise safety issues if it is not handled properly.

### 6.3.4 Final Word

Given a world of infinite bandwidth and no interference, perfect wireless communication and excellent hardware it may be possible to avoid the issues addressed in this work. However, such a world does not exist and probably will not in the foreseeable future. Given the imperfection of digital communication, the chaotic behavior of both hardware and its interaction with software and the network, we have provided a roadmap for achieving effective multi-stream communication with synchronization in an imperfect world.

# References


[1] The American Heritage® Dictionary of the English Language, Fourth Edition, Copyright © 2000 by Houghton Mifflin Company.

[2] M. Fujita, "Digital creatures for future entertainment robotics" IEEE International Conference on Robotics and Automation, 2000, Proceedings. ICRA '00., Vol. 1, pp. 801-806, April 24-28 2000.

[3] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, K. Fujimura, "The intelligent ASIMO: system overview and integration" IEEE/RSJ International Conference on Intelligent Robots and System, 2002., Vol. 3, pp. 2478-2483, Sept 30-Oct 5, 2002

[4] D. Lawrence, "Stability and Transparency in Bilateral Teleoperation." IEEE Transactions on Robotics and Automation. Vol. 9, No. 5 October 1993.

[5] J. Speich, K. Fite, M. Goldfarb, "A Method for Simultaneously Increasing Transparency and Stability Robustness in Bilateral Telemanipulation." IEEE International Conference on Robotics and Automation, 2000.

[6] J. Speich, M. Goldfarb, "Implementation of Loop-Shaping Compensators to Increase the Transparency Bandwidth of a Scaled Telemanipulation System." IEEE International Conference on Robotics and Automation, 2002

[7] H. Flemmer, B. Eriksson, J. Wikander, "Control Design for Transparent Teleoperations with Model Parameter Variation." IEEE International Conference on Robotics and Automation, 2002

[8] G. M. Mair, "Telepresence-the technology and its economic and social implications" Proceedings of 1997 International Symposium on Technology and Society. pp. 118-124, June 20-21, 1997

[9] T.B. Sheridan, "Defining Out Terms", Presence Volume 1, Number 2, Spring 1992, pp. 273-274

[10] T.B. Sheridan, "Telerobotics, Automation, And Human Supoervisory Control" MIT Press, Massachusetts, 1992

[11] NASA Space Telerobotics Program's Real Robots On The Web: http://ranier.hq.nasa.gov/telerobotics_page/realrobots.html

[12] Robin Morphy, "Human-Robot Interaction in Robot-Assisted Urban Search and Rescue." Workshop on Human-Robot Interaction IEEE International Conference on Robotics and Automation. D.C. 2002

[13] Robots developed by San Francisco Robotics Society of America (SFRSA) in response to the September 11 Terrorist Disaster in New York City. http://www.robots.org/WTC911Cover.htm

[14] S. E. Everett. R. V. Dubey, "Model-Based Variable Position Mapping for Telerobotics Assistance in a Cylindrical Environment." IEEE International Conference on Robotics and Automation. Vol. 3. pp. 2197-2202, May 1999.

[15] P.G. Backes, K. S. Tso. And G. K. Tharp. "Mars Pathfinder Mission Internet-Based Operations Using WITS." Workshop on Internet Robotics IEEE International Conference on Robotics and Automation. Pp. 284-291. May 1999.

[16] D. Kwon, K. Y. Woo. H. S. Cho. "Haptic Control of the Master Hand Controller for a Microsurgical Telerobotics System." IEEE International Conference on Robotics and Automation. Vol. 3. pp 1722-1727, May 1999.

[17] M. Tanimoto. F. Arai. T. Fukuba. And M. Negoro, "Force Display Method to Improve Safety in Teleoperation System for Intravascular Neurosurgery." IEEE International Conference on Robotics and Automation. Vol. 3. pp. 1728-1733. May 1999.

[18] M. Ghodoussi, S. Butner, Y. Wang, "Robotic Surgery – The Transatlantic Case." IEEE International Conference on Robotics and Automation. D.C., 2002.

[19] RAMS workstation from NASA's Jet Propulsion Laboratory (JPL) at: http://robotics.jpl.nasa.gov/tasks/rams/accomplishments/robot/ramsTB.html

[20] P.X. Liu, M.Q.H. Meng, J.J. Gu "A teleoperation platform for networked home healthcare robotic system over the Internet" Proc. Of the Second Joint, 24th Annual Fall Meeting of the Biomedical Engineering Society, Engineering in Medicine and Biology, EMBS/BMES Conference, 2002, Vol. 3, pp. 2369-2370, Oct 23-26, 2002

[21] R. Luo, W. Z. Lee, J. H. Leong, "Tele-Control of Rapid Prototyping Machine Via Internet for Automated Tele-Manufacturing." IEEE International Conference on Robotics and Automation, Vol. 3, pp. 2203-2208, May 1999

[22] United State's RQ-1 Predator MAE Unmanned Aerial Vehicle (UAV): http://www.fas.org/irp/program/collect/predator.htm

[23] Gerard McKee, "The development of Internet-Based Laboratory Environments for Teaching Robotics and Artificial Intelligence." International Conference on Robotics and Automation, D.C., 2002

[24] Wilcox, B. Matthies, L.; Gennery, D.; Cooper, B.; Nguyen, T.; Litwin, T.; Mishkin, A.; Stone, H.; "Robotic vehicles for planetary exploration" Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on , 12-14 May 1992 Page(s): 175 -180 vol.1

[25] IEEE International Conference on Robotics and Automation, Internet Robotics Workshop, 2000

[26] K. Brady, T. J. Tarn, "Internet-Based Remote Teleoperation." IEEE International Conference on Robotics and Automation. Leuven. Belgium. May 1998.

[27] R. Riedi, M. Course, V. Ribeiro, R. Baraniuk, "A Multifractal Wavelet Model with Application to TCP Network Traffic." IEEE Transactions on Information Theory (Special Issue on Multiscale Signal Analysis and Modeling). Pp. 992-1018, April 1999.

[28] W. Leland, M. Taqqu, W. Willnger, D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)." IEEE/ACM Transactions on Networking. Vol. 2, No. 1, February 1994.

[29] V. Paxson and S. Floyd, "Wide Area Traffic – The Failure of Poisson Modeling." IEEE/ACM Transactions on Networking, 1995.

[30] J. Beran. R. Sherman, M.S. Taqqu, W. Willnger, "Long-Range-Dependence in Variable-bit-rate Video Traffic." IEEE Transactions on Communication. 1995

[31] M.E. Crovella, A, Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes." IEEE/ACM Transactions on Networking. 1997

[32] J. Gao, I. Rubin, "Analysis of Random Access Protocol under Bursty Traffic." Proceedings of IFTP/IEEE International Conference on Management of Multimedia Networks and Services, Chicago, 2001

[33] J. Perret, C. Prout, R. Alami, R. Chatila, "How to teleprogram a remote intelligent robot" Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and System '94, "Advanced Robotics Systems and the Real World", IROS '94, Vol. 1, pp. 397-404, Sept 12-16, 1994

[34] M.R. Stein, R.P. Paul, P.S. Schenker, E.D. Paljug, "A cross-country teleprogramming experiment" Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems 95. "Human Robot Interaction and Cooperative Robots", Vol. 2,pp. 21-26. Aug 5-9 1995.

[35] D. Pai. "ACME. A Telerobotic Measurement Facility for Reality-Based Modeling on the Internet." IROS Workshop on Robots on the Web. Canada. 1998.

[36] R. Simmons, "Xavier: An Autonomous Mobile Robot on the Web." IROS Workshop on Robots on the Web. Canada, 1998

[37] N. Y. Chong, T. Kotoku, K. Ohiba, K. Komoriya, "Remote Coordinated Controls in Multiple Telerobot Cooperation." IEEE International Conference on Robotics and Automation. San Francisco., April 2000.

[38] L. F. Penin, K. Matsumoto, S. Wakabayashi, "Force Reflection for Time-delayed Teleoperation of Space Robots." IEEE International Conference on Robotics and Automation. San Francisco. April 2000.

[39] M. Stein, "Painting on the World Wide Web: The PumaPaint Project." IROS Workshop on robots on the Web. Canada, 1998

[40] P. Saucy, F. Mondada, "KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet." IROS Workshop on Robots on the Web. Canada, 1998.

[41] Network-Centric Applied Research Team Laboratory: http://ncart.scs.ryerson.ca/

[42] A. Ferworn, R. Roque, and I. Vecchia, "MAX: Wireless Teleoperation via the World Wide Web", Proc. of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering, Edmonton, Alberta, Canada, May 9-12 1999.

[43] A. Ferworn, R. Roque, and I. Vecchia, "MAX: Teleoperated Dog on the World Wide Web", Proc. of the 2nd International Workshop on Presence, The University of Essex, Colchester, UK, 6-7 April 1999.

[44] MAX the Internet Robot: http://max.scs.ryerson.ca/

[45] A. Ferworn, A. Arora, D. Ostrom and W. Shiu, "The Network-Enabled Powered Wheelchair Adaptor Kit - First Prototype", Proc. of the International Conference for Upcoming Engineers (I-CUE'03), Toronto, Canada, May 2, 2003.

[46] A. Ferworn, W. Lu and J. Pham, "Low Bandwidth and Control of a Mobile Robot on a Wireless Personal Digital Assistant", Proc. of the International Conference for Upcoming Engineers (I-CUE'03), Toronto, Canada, May 2, 2003.

[47] A. Ferworn, and W.Lu, "Optimization For Video and Telebotic Control on Palm OS PDAs", Proc. of the International Conference on Internet Computing (IC'02), Las Vegas, USA, June 24-27, 2002.

[48] A. Feworn, W. Lu, J.E. Coleshill and W. Shiu, "WAX -- Next Generation Robot on the Web", Proc. of ht International Conference for Upcoming Engineers (I-CUE'02), Toronto, Canada, May 2, 2002.

[49] A. Ferworn, W. Shiu, and W. Lu, " Constrained Image Understanding Using Lossy Compressed Images", Proc. of the IASTED International Conference for Robotics and Applications (RA'01), Clearwater, Florida USA, Novemer 19-22, 2001.

[50] A. Ferworn, and J.E. Coleshill, "Challenges for Mobile Internet Appliances", Proc. of the IASTED International Conference for Robotics and Applications (RA'01), Clearwater, Florida USA, November 19-22, 2001.

[51] A. Ferworn, B.D. Torres, S. Patel, and S. Kanellis, "The Internet-Enabled Furnace", Proc. of the IASTED International Conference for Intelligent Systems and Control (ISC'01), Clearwater, Florida USA, November 19-22, 2001.

[52] A. Ferworn, W. Shiu, and K. Plataniotis, "Constrained Image Understanding for an Internet Robot Supporting Telepresence", Proc. of the 2001 IEEE Canadian Conference on Electrical and Computer Engineering, Toronto, Ontario, Canada, May 13-16 2001.

[53] Elhajj, I., Ning Xi, BooHeon Song, Meng-Meng Yu, Wang Tai Lo, Yun Hui Liu, "Transparency and synchronization in supermedia enhanced Internet-based teleoperation", Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on , Volume 3, pp 2713 -2718, 11-15 May 2002

[54] Nishino, R., Sezaki, K., "A synchronization mechanism among various media including teleoperation in multimedia communications", Industrial Electronics, Control, and Instrumentation, 1995., Proceedings of the 1995 IEEE IECON 21st International Conference on , Volume: 1 pp 188 -192 vol.1, 6-10 Nov. 1995

[55] Wolf, H.; Froitzheim, K.; "WebVideo. A tool for WWW-based teleoperation", Industrial Electronics, 1997. ISIE '97., [38] Wolf, H.; Froitzheim, K.; "WebVideo. A tool for WWW-based teleoperation", Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on , Volume: 1 , 7-11 July 1997, pp SS268 -SS273 vol.1

[56] Joseph L. Jones, Anita M. Flynn, Bruce A. Seiger, "Mobile Robots: Inspiration to Implementation, 2nd edition", A.K. Peters, 1999, ISBN: 1-56881-097-0

[57] Interactive-C the C complier for Motorola 6811-based embedded systems: http://www.newtonlabs.com/ic/

[58] BTTV: http://bytesex.org/bttv/

[59] Sun's FAQ about sun packages: http://java.sun.com/products/jdk/faq/faq-sun-packages.html

[60] Java Media Framework: http://java.sun.com/products/java-media/jmf/index.jsp

[61] RFC 1889 – RTP: http://www.faqs.org/rfc/rfc1889.html

[62] Y. Ishibashi and S. Tasaka, "A Comparative Survey of Synchronization Algorithms for Continuous Media in Network Environments",LCN 2000, pp. 337-348.

[63] E. Biersack and W. Geyer, "Synchronized Delivery and Playout of Distributed Stored Multimedia Streams", Multimedia Systems 7, pp. 70-90, 1999.

[64] Y. Ishibashi, S. Tasaka and H. Ogawa, "A Comparison of Media Synchronization Quality among Reactive Control Schemes", INFOCOM 2001, pp. 77-84.

[65] Y. Ishibashi and S. Tasaka, "A Synchronization Mechanism for Continuous Media in Multimedia Communications", INFOCOM 1995, pp. 1010-1019.

[66] Y. Xie, C. Liu, M. J. Lee, T. and N. Saadawi, "Adaptive Multimedia Synchronization in a Teleconference System", Multimedia Systems 7, pp. 326-337, 1999.

[67] H. Liu and M. El Zarki, "Delay and Synchronization Control Middleware to support Real-Time Multimedia Services over Wireless PCS Networks", IEEE Journal on Selected Areas in Communications, Vol. 17, No. 9, pp.1660-1672, 1999

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

[69] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, "Robust Internet Video Transmission Based on Scalable Coding and Unequal Error Protection", Image Comunication: Special Issue on Real-Time Video over the Internet, pp. 77-94, Sept. 1999.

[70] D. Loguinov and H. Radha, "End-to-End Internet Video Traffic Dynamics: Statistical Study and Analysis ", INFOCOM 2002. 10. R. Steinmetz, "Human Perception of Jitter and Media Synchroniztion", IEEE Journal of Selected Areas in Communications, vol. 14, pp. 1442-1435, Sept. 1996

[71] R. Steinmetz and C. Engler, "Human Perception of Jitter and Media Synchronization", Internal Report #43.9310, IBM European Networking Center, Heidelberg, Germany 1993.

[72] C. D. Waters, M. Farrell, R. P. Grainger, P. J. Leahy and B. Mellitt, "Dublin Area Rapid Transit" IEE Proceedings on Electric Power Applications, Vol. 135, Issue 3, pp 134-150, May 1988

[73] S. Tasaka, Y. Ishibashi, and H. Imura, "Stored media synchronization in wireless LANs" in Conf. Rec. GLOBECOM'96, Oct. 1996, pp. 1904-1910

[74] ISO/IEC 11172-1: "Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbps, Part 1: Systems," International Standard, 1993

[75] ISO/IEC 13818-1: "Information technology – Generic coding of moving pictures and associated audio information: Systems," Interncation Standard, Nov, 1994 (also, ITU-T Recommendation H.222.0).

[76] ITU-T Recommendation H.223: "Multiplexing protocol for low bit rate multimedia communication." March 1996.

[77] Y. Ishibashi and S. Tasaka, "A synchronization mechanism for continuous media in multimedia communication," in Proc. INFOCOM'95, Apr. 1995, pp. 1010-1019.

[78] S. Tasaka, H. Nakanish and Y. Ishibashi, "Dynamic resolution control and media synchronization of MPEG in wireless LANs," in Conf. Rec. GLOBECOM'97, Nov. 1997, pp. 138-144.

[79] S. Tasaka and H. Imura, "Dynamic resolution control of stored video traffic in a wireless LAN," in Proc. IEEE PIMRC'96, Oct. 1996, pp. 153-157

[80] D. Shepherd and M. Salmony, "Extending OSI to support synchronization required by multimedia applications." Computer Communications, vol. 13, no. 7, pp. 399-406, Sep. 1990

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

[82] Y. Ishibashi, S. Tasaka and E. Minami, "Performance measurement of a stored media synchronization mechanism: Quick recovery shceme," in Conf. Rec. GLOBECOM'95, pp. 811-817, Nov. 1995

[83] L. Lamont, L. Li, R. Brimont and N. D. Georganas, "Synchronization of multimedia data for a multimedia news-on-demand application." IEEE J. Sel. Areas in Commun. Vol. 14, no. 1, p. 264-278, Jan. 1996.

[84] S. Baqai, M. F. Khan, M. Woo, S. Shinkai, A. A. Khokhar and A. Ghafoor, "Quality-based evaluation of multimedia synchronization protocols for distributed multimedia information systems." IEEE J. Sel. Areas in Communication, vol. 14, no. 7, pp 1388-1403, Sep. 1996

[85] T.D. C. Little and F. Kao, "An intermedia skew control system for multimedia data presentation." In Proc. NOSSDAV'92, pp. 130-141, Nov. 1992

[86] Y. Ishibashi, S. Tasaka and T. Okuoka, "A media synchronization mechanism for MPEG video and its measured performance." In Proc. 13th International Conference on Computer Communication, pp. 163-170, Nov. 1997

[87] M. Daami and N. D. Georganas, "Client based synchronization control of coded data streams." In Proc. IEEE Multimedia Systems, pp. 387-394, June 1997

[88] D. Kohler and H. Muller, "Multimedia playout synchronization using buffer level control." In Proc. 2nd International Workshop on Advanced Teleservices and High-Speed Communication Architectures, pp. 167-180, Sep. 1994

[89] K. Rothermel and T. Helbig, "An adaptive protocol for synchronizing media streams." ACM/Springer Multimedia Systems, vol. 5, no. 5, pp. 324-336, Sep 1997.

[90] Y. Ishibashi, S. Tasaka and A. Tsuji, "Measured performcance of a live media synchronization mechanism in an ATM network." In Conf. Rec. IEEE ICC'96, pp. 1348-1354, June 1996.

[91] Y. Ishibash and S. Tasaka, "A media synchronization mechanism for live media and its measured performance." IEICE Trans. Communications, vol. E81-B, no. 10, pp. 1840-1849, Oct. 1998.

[92] F. A. Cuevas, M. Bertran, F. Oller and J. M. Selfa, "Voice synchronization in packet switching networks." IEEE Network, pp. 20-25, Sep. 1993.

[93] M. Correia and P. Pinto, "Low-level multimedia synchronization algorithm on broadband networks." In Proc. ACM Multimedia'95, pp. 423-434, Nov. 1995.

[94] K. Ravindran and V. Bansal, "Delay compensation protocols for synchronization of multimedia data streams," IEEE, Trans. Knowl. And Data Eng. Vol. 5, no. 4, pp. 574-589, Aug. 1993.

[95] S. Ramanathan and P. V. rangan, "Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks." IEEE/ACM Trans. Networking, vol. 1, no. 2, pp. 246-260, Apr. 1993

[96] P. N. Zarros, M. J. Lee and T. N. Saadawi, "Interparticipant synchronization in real-time multimedia conferencing using feedback." IEEE/ACM Trans. Networking, vol. 4, no. 2, pp. 173-180, Apr. 1996

[97] P. V. Rangan, S. S. Kumar and S. Rajan, "Continuity and synchronization in MPEG." IEEE J. Sel. Areas in Communication, vol. 14, no. 1, pp. 52-60, Jan 1996

[98] A. La Corte, A. Lombardo, S. Palazzo and G. Schembra, "A feedback approach for jitter and skew enforcement in multimedia retrieval services," in Conf. Rec. IEEE GLOBECOM'95, pp. 790-794, Nov. 1995

[99] 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, pp. 390-393, June 1996

[100] K. Yamaoka, Y. Sakai, T. Yoshida and A. Ichikawa, "Media synchronization method for motion video based on buffer control." In Conf. Rec. IEEE GLOBECOM'95, pp. 785-789, Nov. 1995

[101] H. Kawai and K. Sezaki, "Adaptive estimation of changing delay pdf for intra-media synchronization." Technical Report of IEICE, CQ97-60, Dec. 1997

[102] S. Treelasanatabron, T. Yoshida and Y. Sakai, "Intramedia synchronization control based on delay estimation by Kalman filtering." IEICE Trans, Communication, vol. E81-B, no. 5, pp. 1051-1061, May 1998.

[103] D. P. Anderson and G. Homsy, "A continuous media I/O server and its synchronization mechanism." IEEE Computer pp. 51-57, Oct. 1991

[104] L. Delgrossi, C. Halstrick, D. Hehmann, R. G. Herrtwich, O. Krone, J. Sandvoss and C. Vogt, "Media scaling in a multimedia communication system." ACM/Springer Multimedia Systems, vol. 2, no. 4, pp. 172-180, Oct. 1994.

[105] M. Katsumoto, N. Seta and Y. Shibata, "A unified media synchronization methods for dynamic hypermedia system." Trans. IPSJ, vol. 37, no. 5, pp. 711-720, May, 1996

[106] S. Tasaka, H. Nakanishi and Y. Ishibashi, "Dynamic resolution control and media synchronization of MPEG in wireless LANs." In Conf. Rec. IEEE GLOBECOM'97, pp. 138-144, Nov, 1997

[107] S. Tasaka and Y. Ishibashi, "A performance comparison of single-stream and multi-stream approaches to live media synchronization." IEICE Trans. Communication, vol. E81-B, no. 11, pp. 1988-1997, Nov. 1998

[108] K. Fadiga, Y. Ishibashi and S. Tasaka, "Performance evaluation of a dynamic resolution control for video traffic in media-synchronized multimedia communication" IEICE Trans. Communication, vol. E81-B, no. 3, pp. 565-574, Mar. 1998.

[109] K. Rothermel and T. Helbig, "An adaptive protocol for synchronizing media streams." ACM/Springer Multimedia Systems, vol. 5, no. 5, pp. 324-336, Sep. 1997

[110] S. T. Liang, P. L. Tien and M. C. Yuang, "Threshold-based intra-video synchronization for multimedia communications." IEICE Trans. Communication, vol. E81-B, no. 4, pp. 706-714, Apr. 1998

[111] M. C. Yuand, B. C. Lo, Y. G. Chen and P. L. Tien, "A synchronization paradigm with QoS guarantees for multimedia communications." In Conf. Rec. IEEE GLOBECOM'99, pp. 214-220, Nov. 1999

[112] L. Li, A. Karmouch and N. D. Georganas, "Synchronization in real time multimedia data delivery." In Conf. Rec. IEEE ICC'92, pp. 587-591, June 1992

[113] K. Taniguchi, H. Tachikawa, T. Nishida and H. Kitamura, "Internet video-on-demand system architecture-MINS." IEICE Trans. Communication, vol. E79-B, no. 8, pp. 1068-1075, Aug. 1996.

[114] W.Y.Chen, "The Development and Standardization of Asymmetrical Digital Subscriber lines," *IEE Commun. Mag.*, pp. 68-72, May 1999.

[115] HandyBoard – http://handyboard.com/

[116] Linux Open Sound System - http://www.opensound.com/

[117] DU Meter – http://www.dumeter.com/