INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A High Performance Audiovisual Communication System:

Aoxiang Xu

Department of Electrical and Computer Engineering
McGill University

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

September, 2000

Abstract

The development of a communications infrastructure for the Shared Reality Environment, a high-end immersive, telepresence space, is described. Such a system must support low-latency, high fidelity, bi-directional audio and video transfer between a number of locations interconnected by an IP network.

As a first attempt, MJPEG video, accompanied by monaural audio was transmitted between two laboratory spaces in the McGill Centre for Intelligent Machines. The relatively high latency resulting from JPEG compression and decompression motivated a hybrid approach in which raw data was selectively transmitted whenever the overhead in doing so was less than the cost of JPEG processing.

Next, to scale up to the demands of transmitting multi-channel audio over greater distances, the protocol was refined and demonstrated by streaming, over the Internet, a live concert from McGill's Redpath Hall to an audience at New York University. This is believed to be the first-ever demonstration of this nature.

While these experiments have proven successful within the limited context of their test environment, some challenges remain to be addressed in order for the system to support the full demands of a Shared Reality immersive telepresence application.

Résumé

Le développement d'une infrastructure de communications pour l'Environnement de Réalité Partagée, un espace d'immersion et de téléprésence à la fine pointe de la technologie, est décrit. Ce système doit soutenir des transferts bi-directionnels d'audio et de video de haute fidélité avec des délais restreints, entre plusieurs espaces connectés en réseau IP.

En premier lieu, du vidéo MJPEG accompagné d'un signal audio monaural est transmis entre deux espaces laboratoires dans de Centre des Machines Intelligentes de l'université McGill. Les délais relativement élevés résultant de la compression et décompression du format JPEG poussent à une seconde approche pour laquelle de l'information non-compressée est transmise, de façon séléctive suivant les coûts relatifs de cette approche par rapport à la première.

Par la suite, pour s'adapter à la demande de transmission d'audio multi-canaux sur des grandes distances, le protocole est raffiné et démontré par la transmission en temps réel sur Internet d'un concert ayant lieu au Redpath Hall de McGill jusqu'à l'université de New York. Cette démontration est la première de son genre.

Alors que ces expériences ont été complétées avec succès dans le cadre limité d'un environnement de test, certains défis doivent être surmontés afin de démontrer l'utilité du système face aux demandes plus élevées d'un système de téléprésence par immersion dans une réalité partagée.

Acknowledgments

I would like to take this opportunity to thank Dr. Jeremy Cooperstock of the Department of Electrical and Computer Engineering, for his supervision of my research project and for bringing me into this wonderful area. His invaluable vision and unending support have always inspired me to make progress in my project. It was always a pleasure and honor to work with him.

I would like to thank Dr. Bruce Pennycook, Dr. Wieslaw Woszczyk, Zack Settel, Quan Nguyen, Geoff Martin, Jason Corey and Gilbert Soucy for all the help I received from them during the multi-channel audio project.

As well I would like to thank Mr. Shawn Areseanu, with whom I shared the office and many laughs during the past two years, and for developing the background removal algorithm, which has been used for the improvement of the Shared Reality Environment audiovisual system. I also thank Ms. Wei Sun for many helpful discussions.

Finally, the love, understanding, and support from my wife and my parents have proved to be a source of strength for me over the years.

Table of Content

Chapter 1: Introduction

1.1 Videoconferencing System - a New Paradigm for Human Interaction

The development of technology that allows people to communicate effectively over vast distances has been one of man's most important achievements. While the evolution from simple communication by smoke and fire to the use of telegrams and telephones spanned millennia, only a mere hundred years separated the latter technology from our present-day global infrastructure. Videoconferencing systems, exchanging both audio and video information, the two principal modalities of human communication, have been in use for decades. They provide the basic framework for humans to interact with each other without requiring participants to be co-located.

The emergence of the Internet has reshaped our lives in many ways, not the least of which has been affording the possibility of low–cost videoconferencing systems to the average user. In early systems, special high-cost equipment was required, but with the emergence of Internet-capable videoconference protocols, PC, modem, audio/video capture device and some sort of videoconferencing software is all that is needed.

Current videoconferencing systems are a step forward in the attempt to allow face-to-face communication in the absence of physical proximity. However, these systems lack the sensory fidelity associated with an in-person meeting. As a result, demanding

applications such as multi-party tutorials, distributed musical performance or remote collaboration on complex medical procedures, are very limited.

1.2 Overview of the Shared Reality Environment

The current design of videoconferencing systems does not make any attempt to compensate for these limitations. Most importantly, the computer, which is responsible for capturing, transmitting and displaying the audiovisual data, has no awareness of the context of activity among the participants. Recent research in human-computer interactions, e.g. ubiquitous computing [1] and computer-augmented environment [2][3], has shown the possibility for the computer to play a more active role in technology filled environment, such as the advanced videoconferencing systems.

The Shared Reality Environment (SRE) is a project that aims to explore the challenging research problems associated with distributed computer-mediated human-human interaction. The intent is to develop technology and investigate new communication metaphors that support highly interactive, complex group activity in a distributed setting. Unlike existing systems, the SRE is highly aware of the context and uses this information to mediate the interaction.

The planned testbed consists of a minimum of three small audio-insulated rooms, each equipped with high-resolution video projectors, cameras, microphones, and multi-channel audio, interconnected by a high performance network. The video will be rear-projected

to cover three walls of each room, thereby encompassing the users' visual field and creating the illusion of a larger shared space. Multi-channel audio will be used to produce effective spatialization of sound sources, enhancing the sense of co-presence. In terms of sensory fidelity, the SRE provides two significant improvements over existing videoconferencing systems: spatialized audio[1] [4] and immersive video.

In conventional videoconferencing systems, each participating site appears on a separate display, or occupies a specific window location, for example, quadrant of a 2 by 2 grid. The SRE, in contrast, must mix the audiovisual streams from multiple sites to give users the impression of co-presence. This involves the synthesis of an appropriate video display on each wall that aligns with its neighboring displays, such that the viewer perceives a single, continuous environment rather than three separate display surfaces.

Human audio perception provides a wealth of sensory information, allowing listeners to locate a particular sound source with a high degree of accuracy and filter out other sources. In the SRE, we wish to reproduce audio such that a user's experience of a symphony performance would offer the same auditory effect as would be possible if the symphony was physically present. This goal necessitates reproducing the effect of audio sources at different spatial locations based on a mapping from desired location to a set of control parameters.

In essence, the core of the SRE is a high performance, audiovisual communication system. This thesis presents the initial efforts of implementing such a system. Chapter 2

[1]Please see www.cim.mcgill.ca/~jer

discusses some technical issues related to the development of an audiovisual environment over best-effort networks, such as the Internet, elaborating on relevant previous research. Chapter 3 describes a prototype high-performance audiovisual communication system developed for the initial SRE tesbed, and explores its limitation. Next, Chapter 4 deals with the development of a robust transmission delivery of high-fidelity, multi-channel audio data over the Internet. In Chapter 5, lessons learned and future directions of research for the SRE communication system are discussed.

Chapter 2: Design Issues of a High Performance Audiovisual Communication System over Internet

In this chapter, issues relating to audiovisual communication system are discussed. First, the desired features of such a system for the SRE testbed are outlined, followed by an introduction to the overall structure of a general audiovisual communication system. The remainder of the chapter presents an overview of previous literature in this area.

2.1 Audiovisual Features of the SRE Testbed

Because the SRE was designed to support applications such as distributed musical performance and remote surgical operation, it sets high standards for the quality of the audiovisual signals being used. As discussed in Chapter 1, the use of immersive video and spatialized audio are two significant improvements of the SRE over conventional videoconferencing systems. Immersive video display requires the video signal to have both high resolution and high frame rate, as a minimum, on par with National Television System Committee (NTSC) video. While stereo audio is the most popular computer audio format, the intended introduction of spatialized sound and the need to support distributed musical performance, demand high-fidelity and multi-channel audio (i.e. more than the two channels offered by stereo). The SRE also places strict constraints on latency, which is the time between the event occurring at the source and its display in another location. Distributed musical performance, in particular, requires very low

latency in order to maintain effective interaction between participants. In general, latency of 50 milliseconds (ms) is considered to be the upper bound for a performance application [5].

2.2 Overall Structure of an Audiovisual Communication System

The basic structure of an audiovisual communication system can be described by the block diagram shown in Figure 2.1. Each block can be further decomposed into a group of hardware or software components that carry out certain functions.

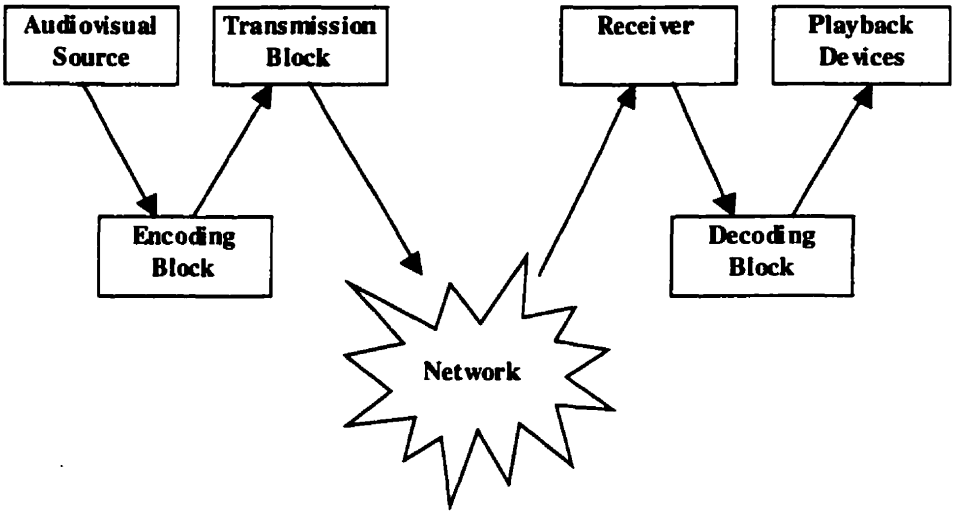


Figure-2.1. Basic structure of audiovisual communication system

In the sending side, the encoding block consists of components that are responsible for sampling the audio/video data, Analog/Digital (A/D) conversion and, if necessary, data compression. In general, sampling and A/D conversion are implemented in hardware, while compression can be implemented in either hardware or software depending on the

requirement of the application. In the following sections of this chapter, the popular audio/video formats and compression schemes are summarized.

The output of the encoding block is passed to the transmission block, which is responsible for transmitting the data to the peer. The transmission block can be well described by the Open System Interface (OSI) seven-layer model [6], which is mostly composed of software components except the modules located in the physical layer. In order to transmit data properly over the network, one has to deal with issues like error correction, flow control and congestion control. Various protocols like Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Protocol (IP), are implemented to deal with these issues, while providing a common foundation for the transmission of data with different purposes. In the OSI model, these protocols are located in the low layers. In real applications, they are usually implemented as part of the operating system. Most of the application specific functions are in the user space.

For the multimedia communication system, due to the large variety of audio/video formats and different performance requirements, it is increasingly difficult to extract the generic properties needed to build a "one-size-fit-all" protocol (e.g. TCP). Others proposed application-level framing (ALF)[7] that encourages the implementation of as much functionality in the application space as possible. In fact, most of the existing videoconferencing systems follow this paradigm [8,9,10,11,12].

The structure at the receiving side is very similar to the sending side. The transmission block receives the data from the network, while the responsibility of the decoding block is the conversion of this data into formats recognizable by the playback devices. In order for the data to be properly delivered, the transmission blocks of the sender and receiver have to cooperate with each other. In the following sections, the issues arising from each function block of the audiovisual communication system, as well as the special requirements imposed by the SRE are discussed.

2.3 Latency of the System

The latency of an audiovisual communication system is the time period between the capture of the actions (audio or video) and their presentation to the remote viewer. It includes sequential time periods spent on the encoding block, the sending transmission block, the network, the receiving transmission block and the decoding block. The latency related with the network depends on two factors: a propagation delay caused by the finite speed of light and latencies in transmission equipments; the transmission delay that depends on the speed of the media (how many bits per second the media can transmit).

Previous research on end-to-end delay is rather limited [13]. Once each component along the path is fixed, the latency is limited to certain values within a fixed range. On the other hand, if there is a predefined constraint on the latency, the choice of components that can be used to build the communication system is also limited.

For the SRE, latency is the most important factor since some of the key applications, like a distributed music performance, has very strict requirements on latency. Studies by telephony companies suggest that human conversation cannot tolerate latency in excess of 200 ms [14]. For musical interaction, this value should be significantly lower. In general, it is believed that a latency of 50 ms is an upper limit for maintaining effective musical interaction [5]. As discussed in the following sections and later chapters, it is very difficult to satisfy this latency constraint when communicating over a wide area network (WAN), even by reducing the time spent in the coding blocks.

2.4 Video Format

The overall quality of the visual effect of a videoconferencing system is controlled by the characteristics of the video frame transmission. Several factors, such as resolution of the video frame, compression format, frame rate, and latency are involved. The scenery is captured and digitized into video frames by a frame grabber. These frames are usually raw images, in which each pixel is described by numbers that represent color intensity. The raw images have the highest quality, and, unfortunately, the highest bandwidth requirement. For the format of RGB 24, each pixel is described by 24 bits. For a video stream frame rate of 30 fps and a resolution of 640 by 480, the total bandwidth requirement is about 221 Mbps. Since 100 BaseT is presently the most popular Local Area Network (LAN) configuration, it is quite obvious that we cannot directly transmit raw images at this level. Various compression algorithms were employed to reduce the

size of the video frame, and subsequently the bandwidth requirements. Some popular formats are Motion JPEG (MJPEG), H.261 and Moving Picture Experts Group (MPEG).

MJPEG is a video sequence of images, each separately compressed with the JPEG (Joint Photographic Experts Group) algorithm. JPEG is an ISO standard on digital compression and coding of continuous-tone still image [15]. The goal of JPEG is to develop a general method for image compression. It uses a discrete cosine transform (DCT) based intra-frame compression to take advantage of the spatial redundancy within an image. There are many commercial hardware codecs as well as a free software codec[1] [16] available.

JPEG provides excellent intra-frame compression. However, for a video sequence, there is also a high degree of temporal redundancy among the neighboring video frames. MPEG (Moving Pictures Expert Group) is the ISO standard for coding moving pictures. Besides DCT based compression, it also uses block-based motion compensation for inter-frame compression. There are multiple versions of MPEG: MPEG-1 [17], MPEG-2 [18], MPEG-4 and MPEG-7 [19]. MPEG-1 and MPEG-2 concentrate on how to store and transmit video signals in the most efficient way. MPEG-4 attempts to provide a standard that supports new ways of communicating, accessing, and manipulating digital audiovisual data. It proposes an object-based paradigm for scene representation. MPEG-7 is a new standard 'multimedia content description interface' that extends today's limited multimedia search capabilities to include more information types. MPEG-1 and MPEG-2 introduce more latency than JPEG due to the intra-frame motion-compensated coding. The typical bit rate for MPEG-1 is about 1.5 Mbps. MPEG-2 is designed to

support applications with high bit rates like near studio video quality (10 Mbps) and HDTV (80 to 100 Mbps), while it also supports a lower bandwidth of around 3 Mbps.

H.261 and H.263 are the compression schemes of the ITU videoconferencing standard [20]. Like MPEG, the H.261 and H.263 encoding algorithms use a combination of DCT coding and motion prediction. H.261 has bandwidths ranging from 64 Kbps to 2 Mbps. The most commonly used bit rate is of H.263 is 64 Kbps.

As discussed in section 2.1, the SRE has very strict constraints on latency. Both MPEG and H.261 (H.263) fail to meet this constraint since the motion prediction-based inter-frame coding introduces extra latency. JPEG only involves intra-frame coding, therefore it does not introduce extra latency. As shown in Chapter 3, MJPEG was chosen as the video format for the initial implementation of the SRE since the relatively low latency justifies its relative low compression rate compared to MPEG and H.261.

Unfortunately as discussed in chapter 3, a video communication system based on hardware JPEG codecs still cannot meet the strict latency constraint required by the SRE. This led to our attempt to develop a hybrid system that transmits both MJPEG and processed raw images.

[1] Please see www.ijg.org

2.5 Audio Format

One of the most important differences between the SRE and presently available videoconferencing systems is the quality of the audio data. In most of the available videoconferencing systems, the demands of the applications typically do not call for high quality audio since voice communication alone is employed.

The SRE, on the other hand, has to deliver high fidelity, multi-channel audio in order to satisfy the needs of applications like distributed musical performance. There are two parameters that control the quality of digital audio: the sampling resolution (bits per audio sample) and the sampling frequency. Musicians and professional audio engineers have strict requirements for audio quality. CD quality audio with a sample resolution of 16 bits and a sample frequency of 44.1 kHz, easily meets the needs of the normal audience. However, it falls short for musicians and audio engineers. In fact, a good musician can easily differentiate the tracks recorded at 20 bits from ones recorded at 24 bits.

A major effort of this thesis was the design and implementation of a system that is able to transmit high fidelity, multi-channel audio over the Internet. We attempted to transmit two types of audio formats: AC-3, a 5.1 channel compressed audio stream [21] and a 6 channel uncompressed audio stream. As discussed in Chapter 4, this was, to our knowledge, the first effort to do so, reliably, over the Internet.

The AC-3 audio coding scheme was developed by Dolby Laboratories to provide high-quality, multi-channel, digital audio for the consumer market. In fact, it has been adopted by high definition TV (HDTV) and digital versatile disk (DVD) for their audio systems. AC-3 encoding scheme compresses a Pulse Code Modulation (PCM) representation requiring more than 5Mbps (6 channels x 48hHz x 18bit) into a 384 kbps bit stream, with excellent audio quality.

Unfortunately, the quality of AC-3 is still not good enough to meet the audio standard of musicians and audio engineers. Rather, uncompressed audio is used in almost all the recording and production sessions. We have developed a reliable transport mechanism for uncompressed audio, requiring approximately 14 Mbps (6 channels at 24 bits with a sampling frequency of 96 kHz).

2.6 Transmission Protocol

2.6.1 TCP

There are two popular protocols used for moving data across the network, TCP [22] and UDP [23]. They are usually implemented as part of the operating system and located in the kernel space.

TCP, a connection-oriented, reliable, and full-duplex protocol, uses an acknowledgement and retransmission scheme to guarantee the delivery of data (i.e. if a segment is not acknowledged by the receiver, the sender will continue to retransmit until the data is

received successfully). Furthermore, TCP ensures an ordered reception of packets by delivering packet p_n only after all previous packets, p_i, $i < n$, have been received.

TCP is Internet friendly since it implements very aggressive congestion control. In the modern fiber connection, the error rate is very low (<0.1%) [24]. Most of the data loss is caused by congestion [24]. Once the TCP sender detects packet loss, it considers that it is caused by congestion on the network and applies one of its two congestion control algorithms: slow start or congestion avoidance. Both will decrease the transmission rate, although slow start is more aggressive. By backing off under the situation of congestion, several TCP streams can effectively share the available bandwidth.

The advantage of TCP is that it simplifies the task of the application. The application only needs to copy the data (video or audio frame) into kernel memory through the system call. The protocol located within the kernel will be responsible for the reliable delivery of the data. Therefore, TCP is well suited to applications such as *ftp*, *telnet*, and *http*. These applications require the data to be delivered to the receiver without any loss, while they have no strict constraint on how long it takes to deliver the data.

TCP is unsuitable for the transmission of real-time media. This unsuitability is related to the characteristics of Internet communication, which is typified by widely fluctuating transmission delays and bursts of packet loss. For example, although the average round trip time (RTT) for data sent between McGill and New York University was in the order of 50 ms, it was not unusual to observe RTTs of several hundred milliseconds during

periods of high network traffic. Such delays can cause TCP to activate its congestion avoidance mechanism [25][26] or lead to timeouts at the sender, substantially degrading performance [26][27]. Packet loss has the same effect.

Although, TCP's performance suffers considerably across a lossy connection, it might be suitable to use in a LAN environment where packet loss and RTT fluctuation are not as severe as in the WAN. TCP is used for the data transmission of the prototype audiovisual communication system for the SRE. When the network is not congested, TCP performs acceptably well.

2.6.2 UDP

For reasons explained in the previous section, real-time applications tend to favor UDP, a connectionless, unreliable protocol with no flow control mechanism. UDP simply continues to pass the data to the low layer no matter whether or not the receiver receives the data. Because of its lack of feedback in terms of network congestion, flow control must be performed in a layer above UDP in order to ensure the proper delivery of the data. UDP also supports multicasting, an important feature for efficient distribution of real time multimedia data.

2.6.3 RTP

RTP (real-time transport protocol) [28] provides end-to-end network transport functions suitable for applications transmitting real-time data such as audio and video over

multicast or unicast network services. Its specification states that "*RTP is intended to be malleable to provide the information required by a particular application and will often be integrated into the application processing rather than being implemented as a separate layer.*" The development of RTP was based on the idea of ALF as an application level protocol that provides a thin layer of transmission control. In fact, applications typically run RTP on top of UDP [9].

RTP does not provide any mechanism to ensure timely delivery or other quality-of-service guarantees (QoS). However, it provides the basic framework upon which such a protocol can be built. All the data are carried in an RTP packet, which has a header and a payload. The header specifies the necessary control information about the payload data. Below are listed important fields of the RTP header. One should refer to the protocol specification for more details.

- Payload type
- Sequence number
- Synchronization source (SSRC)
- Contribution source (CSRC)

The sequence number, payload types, and timestamp provide the basic framework to construct a point-to-point reliable transmission protocol. The SSRC, which identifies the source of a stream of RTP packets, and CSRC, which lists the contributing sources for the payload contained in the current RTP packet, provide further supports for building a control mechanism when multiple participants and interactions among them are involved.

An RTP data packet is designed to carry medium data. In order to build an efficient transmission protocol or control mechanism, more information has to be exchanged between participants. This is carried out by defining an RTP control protocol (RTCP). RTCP defines control packets that are carriers of control information and the rules for exchanging these control packets between participants. An RTCP control packet contains all the necessary information such as: sender reports for transmission and reception statistics from participants that are active senders; receiver reports for reception statistics from participants that are not active senders; available bandwidth calculations and allocation; maintenance of the session members.

Overall, RTP provides an excellent foundation on which to build a reliable, application level transmission protocol. However, since it attempts to accommodate every aspect of real time delivery of media data, it is big and complicated. Therefore, developing a system that is fully compatible with RTP is not trivial. As we were more interested in building a working system that can be used to explore various research questions, we opted instead to build a custom application level protocol. Chapters 3 and 4 describe our efforts in developing such protocols, which satisfy the interests of transmitting multi-channel audio over the Internet. It is not surprising that these protocols follow the basic design principles of RTP as all real time data has similar transmission requirements.

2.7 Error Correction

Since packet loss is inevitable in the Internet, a recovery protocol has to be implemented in order to re-deliver these lost packets. In TCP, reliable data transport is achieved through the use of *stop-and-wait* and *go-back-N* algorithms. However, because of their potentially high latencies, these mechanisms are inappropriate for the transmission of real-time media. We thus consider two other methods for coping with transmission loss: forward error correction (FEC) and automatic repeat request (ARQ).

FEC has been known by those in the field of data communication for decades [29,30]. In FEC, a block code or parity code is applied to a set of k packets to generate a set of $n > k$ packets. The receiver then only needs to receive any k-packet subset of the resulting n-packet block in order to recover perfectly the k original packets

The advantage of FEC is that loss recovery happens entirely in the receiver and the sender does not need to know which packet was lost. Since no retransmission is required, no time penalty is imposed by the process of loss recovery. For this reason, more and more people have become interested in applying FEC as an error recovery method in real-time multimedia communication over the Internet [31,32,33,34,35,36,37].

The drawback of FEC is the increased bandwidth required by the redundant data. Furthermore, its ability to recover lost packets is strictly dependent on the loss characteristics of the underlying network. Specifically, FEC cannot recover from a large burst of packet loss, which we observed frequently during the trials between McGill

University and New York University. For this reason, FEC was inappropriate if one requires absolute recovery of all the lost packets. Although FEC is not used in this thesis, it may be appropriate for the SRE, because of the strict latency constraints imposed by interactive applications that have strict latency constraints. In the final chapter of this thesis, more details are provided regarding the possibility of applying FEC.

In ARQ, the sender explicitly retransmits the packets that are requested by the receiver. Although this introduces a delay of at least three one-way trip times, previous research has demonstrated the effectiveness of ARQ for the transmission of real-time media, especially in the case of multicasting [38]. In ARQ, the receiver only informs the sender about the missing packets. This scheme is also called negative acknowledgement (NACK), which is different from TCP that acknowledges each received packet. NACK greatly reduces back-channel traffic during a transmission session. ARQ is implemented in most systems now available. ARQ is used in the audio system introduced in chapter 4.

2.8 Congestion Control

As stated in earlier sections, the majority of packet loss on the Internet is caused by congestion. When the traffic exceeds the available bandwidth, there are buffer overflows on the intermediate routers. Most modern routers drop packets from the tail of the queue when buffer overflow occurs. Congestion control is the algorithm that reduces the transmission rate according to congestion, thereby relieving the pressure on the network. More importantly, multiple participants can share the available and reduced bandwidth

fairly. As most routers do not offer congestion control capability, it is up to the end-to-end system or program to implement proper congestion control.

The classical congestion control algorithms are the ones deployed in TCP. There are basically two algorithms: slow start and congestion avoidance [24,25]. The TCP sender starts a retransmission timer for each packet that was sent. If there is no acknowledgement received when the timer expires, the packet has to be resent. In the meantime, TCP assumes that the missing packet is caused by congestion and activates slow start. During slow start, the sender starts to send only one packet. Then for each ACK received the sender doubles the number of packets it can send until it reaches a certain threshold in which case congestion avoidance is activated to replace slow start. Therefore, the transmission rate is increased exponentially. Slow start is an aggressive congestion control mechanism since it reduces the size of the sliding window to 1. It is based on the assumption that there is no data flow between the two ends.

Applying slow start for each packet loss is overkill since there still might be some packets going through the network. In order to cope with this situation, a congestion avoidance algorithm was invented. It did not appear in the original TCP and was first proposed by Van Jacobson [24]. It is implemented along with other algorithms like fast retransmission. When the receiver receives out-of-order packets, it sends a duplicate ACK for the most recently acknowledged packet. For example, when packet n is received and acknowledged, the receiver expects packet n+1. If the receiver receives packet n+2, or n+3 instead, it will send a duplicate ACK for the packet n to the sender.

When receiving more than three duplicate ACKs, the sender carries out retransmission immediately instead of waiting for the timer to expire. This is called fast retransmission. Since there is packet loss, congestion control has to be deployed. However, it is a less aggressive algorithm than slow start since the transmission rate is only reduced to about one-half of the value when the congestion occurs. This is based on the assumption that the duplicate ACKs indicate that there is still data flow between the two ends. Once congestion avoidance is activated, further transmission rate increases are allowed to be linear rather than exponential. In congestion avoidance, the transmission rate is increased by at most one packet each ACK. In fact, once the transmission rate reaches a certain threshold during slow start, congestion avoidance is activated to slow down the increase.

The implementation of slow start and congestion avoidance make TCP an Internet friendly protocol. It is perhaps the best compromise between reliability and efficiency. Although unmodified TCP is not the ideal protocol for real-time bulk data transfer, there exist many variants of the basic protocol that are better suited to the task [39,40,41,42].

2.9 Layered Source Coding

Layered coding, also known as hierarchical coding or embedded coding, was first developed for packet audio transmission [43]. It is usually implemented as a part of a congestion control algorithm. In layered coding, a signal is separated into subsignals of various importances in order for them to be coded and transmitted separately. In fact, the data format of packetized voice, as specified by ITU-T G.764 is arranged in layers. This

technique was also extended to video coding and transmission. In layered coding, network congestion will always first affect the subsignals of low importance. Thus, hierarchical coding offers a way of achieving error control by preventing loss of perceptually important information.

The process of signal separation and recombination should be lossless. The common methods for separating a signal into subsignals are bit-plane separation {43}, subband analysis/synthesis [44,45,46,47], and unitary transform [48].

The exact implementation is dependent on many factors such as the type of data (audio or video), timing constraints, bandwidth constraints and quality constraints. For audio applications, bit-plane separation is mostly used. In this case, the most important information consists of the most significant bits of the data, and progressively down through the bit layer, the least important subsignal is composed of the least significant bits. In Chapter 4, we apply a bit-plane separation to the uncompressed multi-channel audio data.

The most important idea of layered coding is that by reducing sampling resolution, it is able to reduce the bandwidth usage of the media stream, while keeping the frame rate constant during congestion. It is significantly different from the congestion control mechanism used in TCP, in which transmission rate (i.e. frame rate for audio and video) is reduced during periods of congestion. Layered coding relieves network congestion by reducing the quality of the transmitted media, while preserving the frame rate at which it

is sent. While beneficial, this is a relatively weak solution, which cannot deal with congestion as effectively as the control algorithm employed by TCP. Many applications have implemented layered coding congestion control [42,49,50,51,52].

2.10 Available Videoconferencing Systems

A large number of audiovisual communication systems have been implemented for either commercial or research purposes. The three most influential systems from a historical perspective are *nv* from Xerox PARC [10], *ivs* from INRIA [11] and *vic-vat-wb* from Lawrence Berkeley National Lab [9]. *nv* does not support audio. It uses a custom coding scheme that is designed for Internet and efficient software implementation. Its video compression algorithm is very similar to H.261 discussed in section 2.4. *nv* supports multicasting and is broadly used in the MBone community. *vic* supports several video compressions, although was originally designed for H.261. *vic* also supports multicasting and is the test bed of several studies that involve the distribution of multimedia data with multicasting. *ivs* supports both audio and video. Video is coded with H.261 and audio is coded with several ITU packet audio formats. Originally, only *ivs* implemented a rate adapting congestion control algorithm based on ARQ. However, research has been carried out on both *ivs* and *vic* that add more sophisticated flow and congestion controls based on FEC and ARQ [8,35,36]. These new algorithms are targeted for multicasting environments.

The MMT (multimedia multiparty teleconferencing) system was implemented in the IBM T.J. Waston Research Center as a research prototype [12]. It uses JPEG as video compression and supports rate based congestion control. It also supports algorithms that can carry out video composition in the compressed domain.

All of the above systems are pioneering works in the development of videoconferencing over the Internet. Therefore, they concentrate on the overall structure of the program. In order to work in most network environments, they do not use audio and video formats that require high bandwidth.

This chapter discussed issues related to building a high performance audiovisual communication system, in particular, those related to the network. The following chapters introduce our initial attempts at building a working system.

Chapter 3: The Shared Reality Communication System

In this chapter, the initial attempt to build an audiovisual communication system for the SRE testbed is introduced. The system, transmitting MJPEG video and CD quality audio, was first implemented between two rooms located on the same floor at the McGill Center for Intelligent Machines (CIM). However, compression and decompression of the MJPEG video frame introduced most latency. In order to reduce this latency, a hybrid system was developed in which MJPEG video frames are replaced by small raw video frames containing only the area around the participant. These frames result from a background removal algorithm that takes original full-sized video frames as input. The background is transmitted in the form of MJEPG at a much lower frequency.

The audiovisual communication system consists of four programs: *asender*, *areceiver*, *vsender* and *vreceiver*. *Asender* and *vsender* acquire the audio/video signals and transmit them to the proper peers over the network. *Areceiver* and *vreceiver* receive the audio/video signals from the network and present them to the users. Following the examples of *vic* and *vat* [9], the audio and video signals are handled by different programs. In the SRE, it is very important to have the freedom to process a specific signal without influencing the others. The disadvantage of this design is that it makes synchronization between audio and video more difficult. However, extensive research has been carried out on the synchronization of separated audio/video signals [54].

3.1 Design of the Video System

Both video and audio programs are implemented on SGI O2 machines. Figure 3.1 shows the structure of the *vsender*. Once started, the main process of the program runs as a daemon, communicating both with the user and with the *vreceiver* or other programs. This latter communication is carried out through sockets and follows the common protocol defined by the reactive room. When there is a request for video, the main process will spawn a child process that captures and transmits the video frames. In the meantime, the main process keeps exchanging control information with all interested parties. By passing feedback or new requests to the child process, the main process is able to control dynamically the video quality delivered to the *vreceiver*. One should realize that it is possible for one *vsender* to handle several video sources at the same time by spawning multiple child processes. However, we do not encourage using a single *vsender* to handle multiple video sources unless it is running on a powerful computer with multiple processors and video codecs.

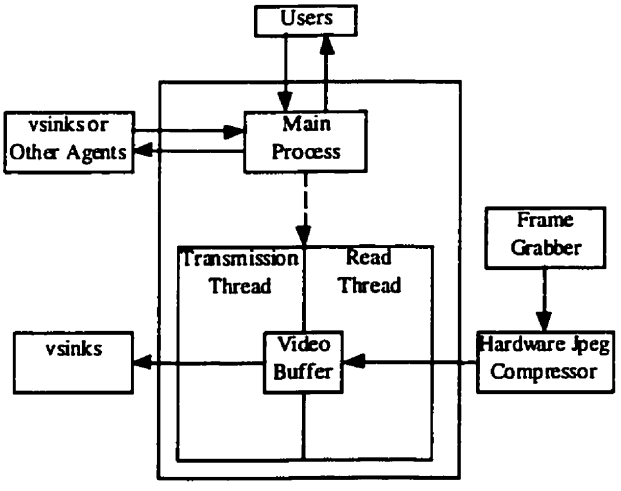


Figure 3.1. Design of the video system

There are two threads in the process. One thread captures raw frames in the format of RGB24 and sends to the hardware JPEG codec built in SGI O2. The other thread transmits the compressed video frames to the network as soon as they leave the codec.

The overall structure of the *vreceiver* is very similar to *vsender*. There are two threads within the process. One is responsible for receiving video frames from the network, while the other forwards these to the hardware codec and displays the decompressed raw images. In order to minimize latency as much as possible, there is no external buffering involved in both the *vsender* and *vreceiver*, only internal buffering by the SGI Digital Media Library (DMlibrary).

We hoped to minimize the compression latency by accessing the built-in hardware JPEG codec on the SGI O2. Furthermore, according to documentation provided by SGI, the O2 machine provides a new platform, the unified memory architecture, which is designed to provide the best performance for any video-related work. Accompanying the special hardware design is the DMlibrary that provides a framework for developing video-related programs.

Since threads share the same process space, the JPEG frame stored in the DMbuffer can be sent directly to the peer by the transmission thread. As stated before, the O2 machine was chosen in the hope that its special hardware and software design would help improve the performance of our program. Unfortunately, as shown in a later section, even using

this hardware codec, the performance still cannot satisfy the requirements of some SRE applications.

TCP is selected for transmitting the video and audio data from sending program to receiving program. This might be a surprise since we have drawn the conclusion in Chapter 2 that TCP is not suitable for real-time multimedia communication. However, the current system was developed for two rooms located on the same LAN. With the 100 BaseT connection, the network performance is typically sufficient, and we seldom experience severe congestion that can cause large packet loss or delay.

Since TCP transmits data in byte stream, all the video frames are transmitted in a user-defined data packet. The data packet includes a header and a payload. The video frame is carried in the payload, while the header includes information such as packet type, sequence number, timestamp, and payload size. One should realize that the design of our data packet format is independent of the RTP protocol, although they are very similar. It is part of the common communication protocol of the reactive room [53], which was established before the final proposal of RTP.

Several modifications were applied to TCP in order to improve its performance. For example, we increased the sender and receiver buffer of the TCP connection, which has resulted in increased throughput [55,56,57].

Although TCP is used here, one should realize that the current system is still in the prototype phase, with modifications planned for the near future. For example, it is our intention to implement multicasting so that one video stream can be distributed to several receivers in the most efficient way. This requires using UDP as the transmission protocol since TCP does not support multicasting.

3.2 Design of the Audio System

The structures of the *asender* and *areceiver* are almost identical to the *vsender* and *vreceiver*. CD quality audio (44.1Khz, 16bit) is transmitted. Each data packet includes about 2ms audio data, which is a much smaller segment (400 bytes) compared to the video frame. TCP, by default, tries to avoid sending too many small messages. This is implemented by the Nagle algorithm and the Delayed ACK [56]. The Nagle algorithm specifies that no data smaller than one TCP segment should be sent if there are still unacknowledged data in the sending buffer. Delayed ACK specifies that the receiver should delay the transmission of ACK of a segment in the hope of piggybacking it with other data. To improve performance, both the Nagle and Delayed ACK algorithms were disabled in the audio system.

3.3 Performance

For a JPEG frame with a resolution of 640 by 480, our video system can sustain an average frame rate of 26fps, which is close to the NTSC standard. However, the key factor that we want to minimize is the latency of the system. Figure 3.2 shows that the average latency for each video frame is very close to 50 ms, even in a LAN environment. The latency of the system includes three major parts: compression, transmission decompression. The time spent on capture and display is relatively minor and can be ignored.

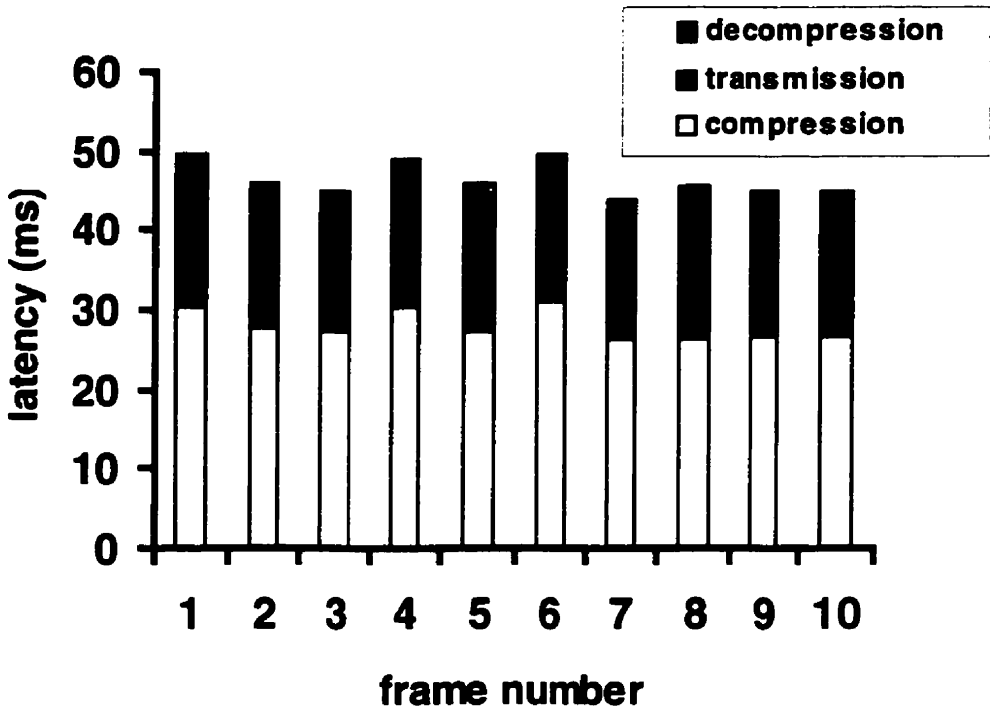


Figure 3.2. Latency of the video transmission

Unfortunately, the latency of this system cannot meet the stringent requirements of an application such as distributed musical performance in which a delay of 50 ms is assumed

to be an upper limit for maintaining effective interaction. Figure 3.2 is generated in a LAN environment where the network latency related with is very low (<5ms). In a WAN, this latency could be as high as 40 to 50 ms, for example, across the continent. The average RTT between McGill and www.ucla.edu, as measured by "ping"[1], is about 40ms.

Since the propagation delay is negligible in the LAN environment, most of the time is spent on compression and decompression. The hardware codec we used is able to support a compression rate of 30fps or one frame per 33 ms. Unfortunately, the codec consumes about all of the 33 ms to do the compression. To the best of our knowledge, currently available hardware codecs have a compression rate of 30fps and a decompression rate of 60fps. This means that a delay close to 50 ms is inevitable if one wants to use JPEG format. Considering that JPEG has one of the shortest latency among the available video compression schemes, the conclusion is that uncompressed video has to be used if a videoconferencing system is to meet the strict latency requirement imposed by tightly synchronous applications such as remote music practice.

3.4 A Hybrid System

We believe that the latency of our implementation is close to the limit that one can achieve with off-the-shelf components, using a compressed video format. Unfortunately,

[1] *ping* measures the round trip time between two sites by sending ICMP echoing packets.

this latency is still too high for latency-sensitive applications. For this kind of application to be carried out effectively, the latency needs to be at the level of 20 to 30 milliseconds. However, in our system the compression and decompression process alone introduces about a 50ms delay, well in excess of current off-the-shelf JPEG codec compression times. Using a specially JPEG codec may be possible, but would be prohibitively expensive for our needs.

Our solution to reduce compression latency is to transmit raw images. This avoids the time spent both on compression and decompression. The other advantage of transmitting raw video is that it becomes much easier for the receivers to manipulate (mix, merge...) the different video streams.

However, transmission of raw images is limited by the available bandwidth of the network. For example, a stream of RGB24 raw video with a resolution of 640 by 480 at 30fps requires 221Mbps. This exceeds the limit of 100 BaseT (Ethernet at 100 Mbps). Although Gigabit Ethernet is able to handle this kind of traffic, it still cannot cope with the multi-stream demands of the SRE. For example, for five streams, the total bandwidth is 1105 Mbps, which exceeds the sustained bandwidth of Gigabit Ethernet.

In most SRE applications, the participants do not carry out actions involving large movements such as running. Also at this early stage, we assume only one participant in each location. Furthermore, the background at each location is relatively static. In fact, we might want to use a synthesized background in some applications. Based on these

assumptions, we propose an image-processing algorithm to the captured raw images, which extracts a bounding box around each participant in the format of raw image data, but requiring less bandwidth due to its reduced size. By transmitting these reduced video frames, we are able to avoid the time spent on image compression and decompression without overloading the underlying network. For example, if the bounding box is 160 ×120 pixels, the bandwidth requirement is reduced to about 14Mbps, permitting 100 BaseT LAN to handle up to 4 video streams at this bandwidth. The background of each location can be transmitted in JPEG format at a lower frequency. Therefore, it is a hybrid system in which both processed raw video frames and MJPEG video frames are transmitted.

There are several problems with the new system. First, the image-processing algorithm introduces an extra delay. However, the time spent on processing the image is smaller than the time required by the process of compression-decompression. At present, the algorithm takes approximately 20 ms on a Pentium III 500 MHz machine to generate a bounding box of 160×120 from a 640× 480 raw image, which is much less than the 50 ms required by the compression and decompression on a SGI O2. A second problem is that if the participant is too close to the camera, the size of the generated raw image might not be significantly smaller than the original raw frame since the participant could easily occupy the entire image. A third problem is that the algorithm can only isolate a single participant at present. While acceptable at this early stage, the long-term goal is to support multiple participants at each location [58].

Figure 3.3. Screen capture from hybrid video agent

Figure 3.3 illustrates a scene produced by this system, employing the background removal algorithm developed by Arseneau [59]. The bounding box can be put into both a natural scene and synthesized background. Since our interest is simply to prototype a working system, no effort was spent on the efficient display of the image.

3.5 Lessons Learned

In this chapter, a simple audiovisual communication system for the SRE testbed was prototyped. An important lesson learned is that compressed video is not suitable for applications that have strict constraints on latency. Our results show that MJPEG compression and decompression introduces significant latency when current off-the-shelf hardware codecs are used.

Raw image data offers more flexibility and imposes no extra delay. Therefore it is more suitable for low latency applications. The disadvantage of raw image is that it requires significant bandwidth. Although the rapid growth of the network industry has already made it possible to transmit a single raw video stream with off-the-shelf equipment (Gigabit Ethernet), it is still too expensive to accommodate multiple raw video streams (as 10 Gigabit Ethernet would be required). Based on the assumption that the background of each location does not change very often, we propose a hybrid system that only transmits part of the raw image by removing irrelevant background with an image processing algorithm. This design demonstrates great potential since it provides both low latency and flexibility for video processing on the receiving end.

In the interests of expediency of implementation, we use TCP as the transmission protocol, which greatly reduces the complexity of the program. Although our system does not meet all the requirements of various applications, it can still be used for experiments that study human-human interaction in our computer-mediated environment.

Chapter 4: Real-Time Internet Streaming of Multi-channel Audio

The previous chapter introduced our initial efforts of building a prototype audiovisual communication system. Although it works nicely in a LAN environment, the system is not equipped to handle real world application over the Internet, where packet loss is quite common. It is very difficult for the system to deliver data in a timely fashion since it uses unmodified TCP as the transmission protocol. In this chapter, we turn our attention to the development of a robust communication system capable of streaming high quality, multi-channel audio over the Internet.

4.1 Introduction

Today, real-time Internet music streaming, such as that available from RealNetworks (realnetworks.com) and MP3 (mp3.com), is regularly enjoyed by millions of listeners. The limiting factor in such transmissions is typically the available bandwidth of the underlying networks. As a result, Internet audio streaming technologies are invariably characterized by relatively low-bandwidth, low-quality, stereo sound. Although this may satisfy the minimal requirements of casual audiences, it is by no means sufficient for a society of musicians and professional audio engineers [60], to whom high-fidelity, multi-channel audio is a necessity. As stated in Chapter 1 and Chapter 2, the SRE is designed to support spatialized audio and the applications like distributed musical performance.

All of these require the delivery of high quality, multi-channel audio. Fortunately, the recent emergence of advanced research networks provides an opportunity to explore the possibilities of real-time streaming of high quality, high bandwidth, and multi-channel audio over the Internet.

We design a system that is able to transmit multi-channel audio over Internet. Using this technology, we carried out two successful demonstrations, one for the 107th Audio Engineering Society (AES) Convention in New York and the second for the 1999 Canarie Advanced Networks Workshop in Toronto. Considering the emergence of Megabit ADSL and cable-modem links to the home, we note that these trials are not merely fanciful research experiments, but rather, likely prototypes of the next generation of consumer entertainment delivery mechanisms.

4.2 The Demonstrations

The first demonstration, on September 26, 1999, involved the transmission of AC-3 audio accompanied by a separate MPEG-1 video stream, requiring a total bandwidth of approximately 3 Mbps. A live performance of the McGill University Swing Band, playing at McGill's Redpath Hall, was delivered to an audience at the Cantor Film Center of New York University, with a time delay of approximately three seconds.[1] The second

[1] A highly conservative 15-second delay was employed during the first half of the performance, but this was later reduced to three seconds as the authors attempted to demonstrate the importance of buffering. However, due to its robust retransmission algorithm, the system continued to deliver lossless audio, despite competing traffic on the network, even as the accompanying MPEG-1 stream exhibited occasional drops.

demonstration, on November 28, 1999, delivered from McGill University to the Metropolitan Toronto Convention Center, repeated the setup used in New York, but decreased the time delay to approximately one second, thereby illustrating the potential of the technology for reliable, real-time transmissions. At the same time, we generated an additional 12 Mbps of competing traffic between the two sites to test the system's robustness to congestion. An additional experiment, employing six channels of uncompressed, linear coded audio, 24 bits per sample at 96 kHz, could not be carried out due to limitations of the hardware.

4.2.1 Hardware

Figure 4.1 illustrates the experimental setup. The output of a mixing console at McGill University was fed into a Sony PCM-800, which converted the six channels of analog input to PCM at 16 bits per channel, 48 kHz, which were in turn provided to a Dolby DP569 encoder via three AES/EBU streams. The resulting AC-3 data, encapsulated in an AES/EBU stream at 1.536 Mbps, was read by the *sender* program running on a Silicon Graphics Indy (R4600 with 64 Mbytes RAM). The sender was responsible for segmenting the audio data into discrete packets and transmitting them over the network, to another Indy R4600 with 150 Mbytes RAM), located at the Cantor Film Center (or the Toronto Convention Center). A *receiver* program, running on this machine, read the packets from the network, extracted the audio data, and provided it via the Indy's AES/EBU port, to a Dolby DP562 decoder, which decompressed the AC-3 data and delivered the audio to a playback system.

The setup for the uncompressed audio experiment called for the six channel audio sources to be connected to A/D converters manufactured by Data Conversion Systems (dCS), whose outputs are integrated by a prototype APX encoder unit. The encoder packs the resulting audio stream into a flexible data format, discussed in the following section, which is then read by the *sender* program running under Linux on a Pentium III PC. While the initial design called for a high speed parallel port interface between the APX and the PC, we found that the parallel port I/O consumed all available clock cycles, leaving no time for the transmission (or reception) of data through the network card, nor the servicing (or generation) of retransmission requests. The next version of this hardware, currently in development, will replace the parallel port interface with a Firewire (IEEE 1394) port, thereby allowing for lower I/O overhead. Although hardware limitations have so far prevented us from demonstrating an end-to-end system in operation, simulations indicate that our system is able to meet the network demands of the uncompressed format.[1]

The sender program is responsible for transmitting the data over the Internet to a receiver program, running on an identical PC, where the audio data is extracted and delivered to an APX decoder, which unpacks the data and passes the streams to D/A converters for playback.

[1] It is our intention to carry out a demonstration of this system at the upcoming 109th AES convention.

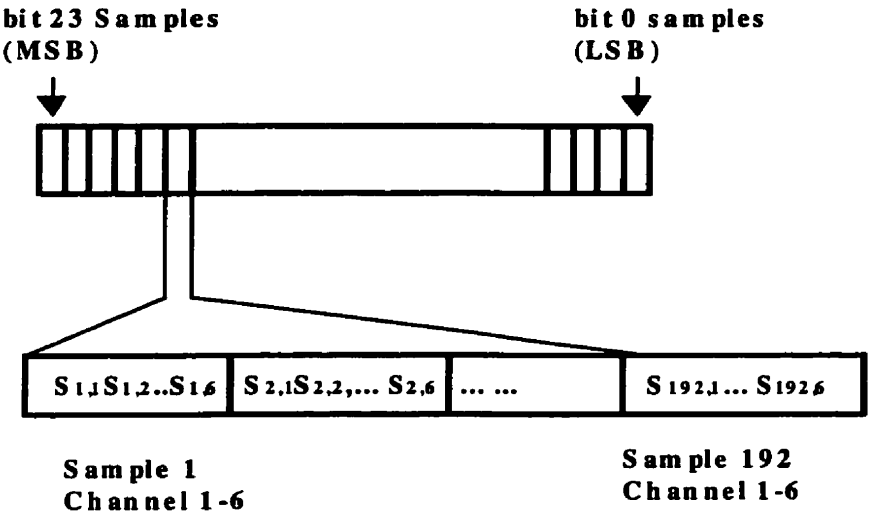


Figure 4.2. Format of an uncompressed audio data block. Every six-bit segment contains the values of one particular bit position of one audio sample over all six channels. The data is ordered with the most significant bit of every sample appearing before the next significant bit. This permits signal reduction without complex manipulations through a simple truncation of the block at any chose resolution.

4.2.2 Audio and Video Format

The AC-3 audio data (encoded Dolby 5.1) is encapsulated in a stereo AES/EBU (48 kHz, 16 bit) stream provided by the Dolby Encoder. AC-3 carries five full-range channels and one lowe-frequency effect channel (subwoofer) [21], but because of compression, does not require the full bandwidth of AES/EBU. While it is therefore possible to reduce the AC-3 bandwidth demands by discarding the padded zeros of the AES/EBU stream, we chose not to do so, as our research goal involves higher bandwidth applications, such as the transmission of uncompressed six-channel PCM data.

For transmission of uncompressed audio data, the APX encoder reads all six channels of data from the A/D converters and packs them into custom data blocks, pictured in Figure 4.2. Each block includes 144 samples of the six-channel audio data, arranged in layers of successively decreasing sample resolution. In this manner, the first layer consists of the most significant bit of each sample for each of the six channels. This is then followed by the second layer, which contains the next most significant bit of each sample, and so on. This is actually an implementation of layered coding that was discussed in chapter 2. This layered coding format is very similar to the method of packetized voice, as specified by ITU-T G.764, designed for the convenience of congestion control. Further details concerning the effect of this layering on system performance are discussed in sections 4.5.4 and section 4.6.4 of this chapter.

In addition to the audio stream, we utilized Cisco's IP/TV system for the encoding, transmission, and decoding of MPEG-1 video at approximately 1.5 Mbps and MPEG-2 video at approximately 3.0 Mbps[1]. As IP/TV does not yet provide support for external synchronization, nor the transport of high fidelity, multi-channel audio, the alignment of audio and video streams was performed manually, immediately prior to the demonstrations. It should be noted that MPEG-2 Advanced Audio Coding (AAC) supports synchronized playback of multi-channel audio with greater fidelity than AC-3, but this format is not currently supported by IP/TV.

[1] Due to limited processor resources at the time, we only utilized the MPEG-1 stream for these demonstrations.

4.2.3 Network Route

For the first demonstration, the audio and video streams were transmitted over the high-performance networks managed by Canarie (Canada) and Internet2 (US) corporations. The network route used during this demonstration, along with the available bandwidth of each link, as measured by *pathchar*[1], is illustrated in Figure 4.3. For the second demonstration, the network route was confined to that portion of the original path between Montreal and Toronto.

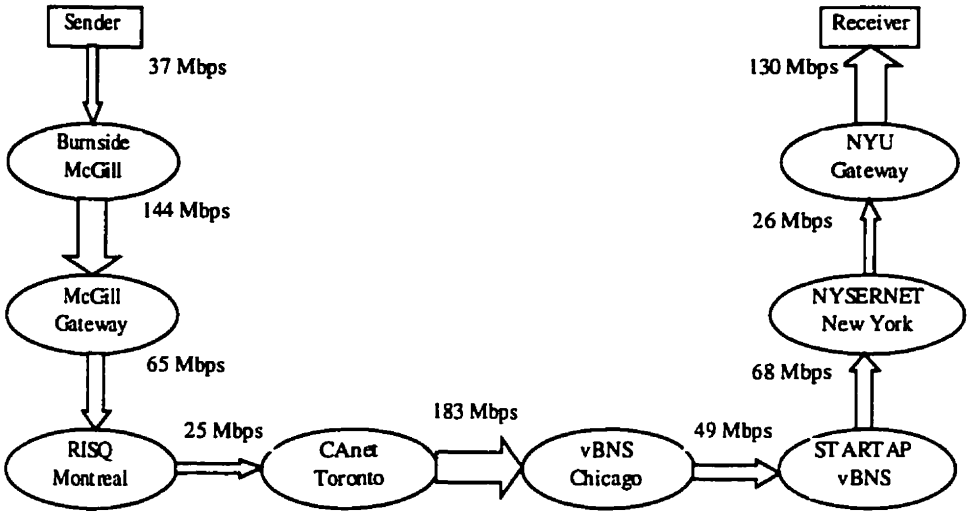


Figure 4.3. The network path between *sender* and *receiver*

Although these networks have relatively high capacity, there were several bottlenecks that limited the available bandwidth. For an AC-3 audio stream, these bottlenecks may not appear significant, as our bandwidth demands amounted to only a fraction of the available capacity. However, transient network congestion induced by other high

[1] Note that there has been considerable discussion as to the accuracy of measurements provided by this tool.

bandwidth applications would prove fatal to a simple audio transmission protocol. We were challenged to ensure robustness, despite the lack of bandwidth guarantees and the unavailability of any bandwidth reservation protocols on the network. Just as on a public Internet, we had to compete with all other traffic, while ensuring the delivery of our AC-3 audio stream.

4.3 The Audio Transmission System

The design of our transmission protocol was the most critical component of the system. Since the audio data was to be streamed and played back in real time, the protocol had to ensure delivery of data to the receiver with low latency, low error and loss rate, and most importantly, without breaks in continuity. It is this final point that ultimately determined the success of the demonstration, as any interruption of the output stream would cause an immediately noticeable break in the music.

Our design was based in part on the Adaptive File Distribution System (AFDP) [61], which provides efficient and reliable delivery of a file to multiple hosts on an Internet. The fundamental difference between AFDP and our application is that the former is not bound by real-time constraints. For a streaming audio application, we are willing to trade a small amount of reliability in exchange for real-time performance.

4.3.1 Transmission Protocol

Based on the discussion in chapter 2, we choose UDP as the main protocol to transmit audio data. However, as demonstrated in later sections, TCP, although limited by the aggressive congestion control, could be useful in some situations.

RTP (real-time transport protocol) [28] provides an application framework for developing program that transmits real-time data such as audio and video over multicast or unicast network services. In fact, applications typically run RTP on top of UDP [9]. As time was limited (we need to deliver a working system for the demonstration deadline), rather than implement the entire RTP specification in our prototype, we opted to develop a lightweight version, tailored for our reliable unicasting needs. Our implementation includes most of RTP's basic services, including payload type identification, sequence numbering, timestamping and delivery monitoring.

4.3.2 Program Overview

Figure 4.4 presents the structure of the sender and receiver programs and illustrates the interactions between them. The main program structures are the same for both the AC-3 and uncompressed versions of the program. At each host, three network sockets are created for communication between the two programs:

A TCP connection initiated by the receiver, used to request transmission of audio data from the sender.

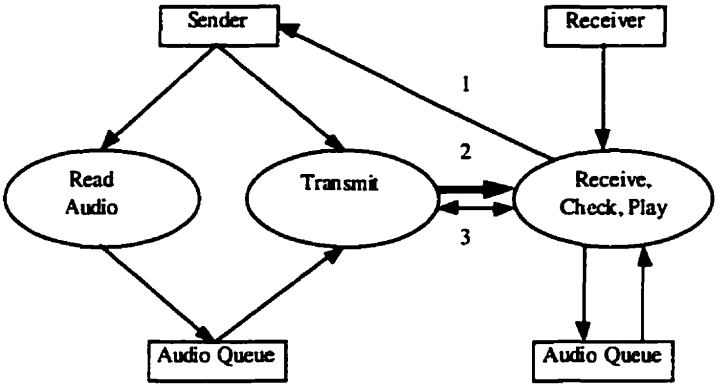


Figure 4.4. Communication software structure. 1 and 3 is TCP;2 is UDP

A UDP channel, used for the transmission and retransmission of the audio data.

A TCP connection initiated by the receiver, used for the exchange of control packets between the two programs and for the occasional retransmission of missing audio data, as will be explained below.

The sender process maintains an audio queue containing pointers to the data packets being transmitted. This process has two threads. One thread continuously reads the output of the audio encoder, stores this data in packets, and adds them to the audio queue. The second thread is responsible for the transmission and retransmission of audio data to the receiver.

The receiver process consists of a single thread, which continuously checks the UDP and TCP sockets for incoming data and stores these received packets in an audio queue. At every iteration through the loop, a periodic timer is checked. Whenever the timer

expires, audio data is dequeued and sent to the playback device and the remainder of the queue is checked for missing data. If necessary, a retransmission request is then issued to the sender.

All the data transmission by our system takes place in the form of user-defined data and control packets. Data packets, normally sent by UDP, except during retransmissions, consist of an identifying header and a payload, which carries audio data. The identifying header includes sequence number, timestamp and the payload data type. It also includes the most recent average transmission rate at the sender. This can be used by the receiver to estimate the current network congestion. Control packets, sent by TCP to provide guaranteed delivery, are used primarily for retransmission requests and exchanging monitoring information between the sender and receiver.

4.3.3 Data Buffering

As discussed in the following sections, retransmission is used for the recovery of lost packets as needed to ensure reliable end-to-end transport. This requires that audio data be buffered at both the sender and the receiver. Both ends maintain the audio data in a ring buffer, implemented as a linear array of pointers to the data packets, in order to allow direct access to the corresponding data segments.

As each block of audio is read at the source, the sender stores this data in a new packet and inserts a pointer to it into the appropriate cell, indexed by sequence number. When the head of the ring buffer reaches the tail, the oldest audio packets are freed, leaving space for newly acquired data. In this manner, the send buffer is always kept full,

allowing for a maximal retransmission window, apart from a short period at the beginning of the program.

At the receiver, the incoming data packets are saved and a pointer to each is added to the buffer in a cell determined by the packet's sequence number. Once the number of data packets stored inside the buffer reaches the playing threshold, this data is dequeued and sent to the playback device. At this point, a periodic timer is started, and on each expiration of the timer, the oldest packets are dequeued from the ring buffer and played. For a lossless network, the receiving rate should be equal to the playback rate and hence, the number of packets in the receive buffer would be constant. However, in the real world, the receive buffer fluctuates because of packet loss caused by network congestion.

4.4 Packet Recovery and Congestion Control Mechanisms

Since packet loss is inevitable, we require a lightweight recovery protocol in our program. This protocol must be reliable, subject to best-effort limitations, and fast, such that the receiver recovers the missing data in time for playback. Our ability to satisfy these demands is determined jointly by the network architecture, link capacity, competing traffic, and the algorithms employed. It is only this last variable over which we have any control.

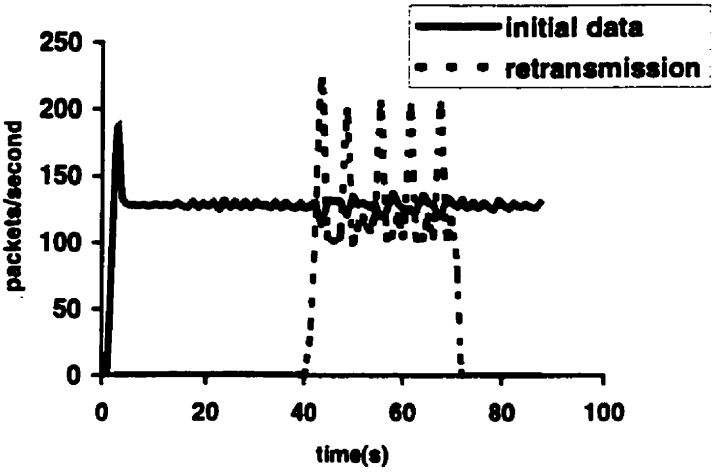
As discussed in chapter 2, there are two basic loss recovery schemes: FEC and ARQ.

FEC was not used in this system since it cannot recover from a large burst of packet loss, which we observed frequently during our trials (see Section 4.5.2). An additional relevant factor is that we were only concerned with reliable, one-way transmission in this application, hence, there was no need for low-latency interactivity.

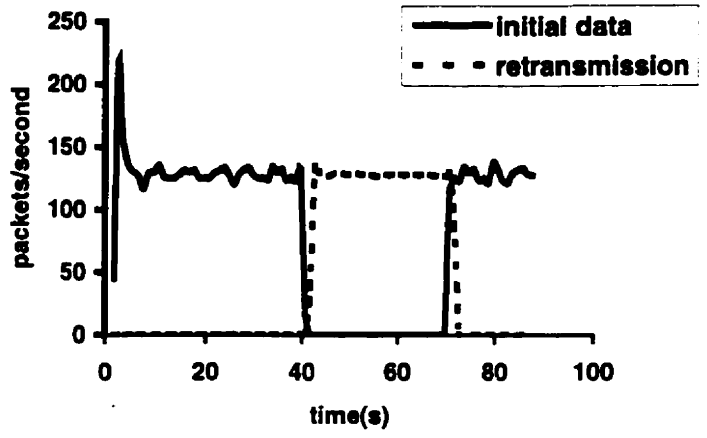
ARQ has been used in our system. It is proved to be quite effective provided that network capacity was not saturated. Although we have so far demonstrated unicast alone, our system can be extended easily to support multicast, permitting its application to the distribution of a concert to several sites at once. It is discussed in chapter 2 that ARQ can be very effective for multicasting application.

Our implementation of ARQ employs a selective-repeat technique. Each data packet is labeled by the sender with a unique sequence number. The receiver detects missing data by checking periodically for a gap in the sequence numbers of received packets. When such a gap is found, the receiver issues a negative acknowledgement (NAK) to the sender as a retransmission request [61].

We illustrate the ARQ-based packet recovery protocol by intentionally dropping 30 seconds of audio data in the middle of an AC-3 transmission, as shown in Figure 4.5. At the onset of simulated packet loss, the receiving rate drops to zero. The receiver then issues a retransmission request and shortly thereafter, the retransmitted packets begin to arrive.



(a)



(b)

Figure 4.5. Demonstration of the loss recovery protocol, in which the receiver starts to drop packets at 40 seconds. (a) The sender begins transmitting packets normally, but in response to a resend request from the receiver, the retransmission protocol is activated at t=40s. (b) The receiver fails to receive any packets via the UDP socket so issues a retransmission request shortly after the onset of packet loss.

Since the packet loss in this case is not caused by network congestion, the simulation is, of course, completely artificial, and serves only to demonstrate the high-level operation

of the program. In practice, retransmission of lost data cannot be guaranteed to arrive in a timely manner during periods of network congestion. We address this point in section 4

4.4.1 Retransmission Protocol

Having decided on the use of ARQ, we were then faced with the choice of retransmission protocol for lost audio packets. Based on the discussion of chapter 2, UDP would seem to be an obvious candidate. However, because of its inherent unreliability, we must also apply our loss recovery protocol to the retransmitted data. Furthermore, the temporal constraint that each audio packet must be received before its playback time mandates that losses of the resent packets are detected quickly and that further repeated retransmissions be carried out with minimal latency.

For the high bandwidth, uncompressed audio data, a UDP based retransmission protocol is used. Once a NAK is issued for missing packets, a timer is started. When this timer expires, the missing packets are rechecked, and if any are still missing, another NAK is sent. This process could continue until it is too late to receive the missing packets. However, a disadvantage of this method is that multiple retransmissions will increase network load, thereby leading to further packet loss. In order to prevent overloading the network with repeatedly resent data, we abort our retransmission efforts on any given packet after three attempts.

Under conditions of high congestion, our experiments demonstrate that TCP is unable to retransmit lost data effectively, due to its network-friendly congestion control algorithms.

For greedy, high bandwidth, real time applications, UDP is thus the only choice for retransmission. However, for low bandwidth audio data such as AC-3, we find that TCP is well suited to deal with retransmissions, as the risk of delay associated with its congestion avoidance mechanism is preferable to continued packet loss under UDP. The reason for this hypothesis is simple: Even if the packets resent by TCP do not arrive in time for playback, the continuing UDP transmission of later audio data will not be affected. Thus, a break in the music, if one is forced to occur, will be of minimal duration. Our hypothesis was confirmed by experiments, described in section 4.5.3, which compare the performance of TCP and UDP for loss recovery of an AC-3 transmission under different levels of congestion. This led us to choose TCP as the retransmission protocol for the AC-3 demonstrations.

4.4.2 Naïve Congestion Control

Retransmitted data naturally increases bandwidth utilization. For the transmission of uncompressed audio at 13 Mbps, some of the network links near the sender approached saturation. Under these conditions, it was thus dangerously easy to overload the network. What happens in this situation can be described as a positive feedback loop: A competing data stream causes congestion, which leads to packet loss in our application. Retransmission is then requested to recover the lost packets. If the competing data stream persists during the retransmission, congestion becomes even more severe, leading to a further increase of packet loss, and so forth, eventually leading to congestion collapse.

In order to avoid this scenario, we define certain thresholds on the network traffic parameters. Once these thresholds are exceeded, the program simply ignores any packet loss. Unfortunately, this method is rather naive. It may activate too early and drop packets that are potentially recoverable or activate too late to prevent congestion collapse. One would prefer a smoother response to congestion.

As discussed in chapter 2, TCP implements an effective congestion avoidance algorithm. It will back off and decrease the transmission rate whenever congestion is encountered. While the nature of our application does not lend itself to a strong congestion control procedure, it would be disastrous to do without one altogether. Therefore, we chose to implement a weaker algorithm, based on layer coding congestion control.

4.4.3 Layered Coding Congestion Control

As introduced in chapter 2, Layered coding congestion control is a common method used in real time multimedia streaming. The idea is to reduce sampling resolution, and hence, bandwidth usage of the media stream, while keeping the frame rate constant, when congestion is encountered. It is significantly different from the congestion control used in TCP, in which case the transmission rate is reduced during periods of congestion. Layered coding relieves network congestion by reducing the quality of the transmitted media, while preserving the frame rate at which it is sent. While beneficial, we note that this is a relatively weak solution, which cannot deal with congestion as effectively as the control algorithm employed by TCP.

Applying the layered coding technique to the uncompressed audio stream, we can drop the n least significant bits of each audio sample to reduce its bit rate, where n varies monotonically with observed congestion. For example, we can reduce the sample resolution from 24 to 16 bits for a 33% decrease in bandwidth requirements. Since the least significant bits are arranged at the end of each audio block, as shown in Figure 4.2, congestion control is easily accomplished by truncating data packets at the desired resolution.

The dynamic variation of sample resolution is the most important component of the congestion control algorithm, and depends on many parameters. In our experimental trials between McGill and NYU, we found that a packet loss of 10% represents a reasonable threshold at which to assume possible network congestion. Therefore, as soon as this threshold is reached within any window, we reduce sample resolution by one bit. For every further 4% increase in packet loss, we drop an additional bit, which is equivalent to reducing the original 24-bits/sample stream by $1/24 \approx 4\%$. In section 4.5.4, we present some measurements demonstrating the effectiveness of the layered coding congestion control in improving packet loss recovery and relieving network congestion.

4.5 System Performance

The characteristics of the underlying network are very important in determining the design of our system and its resulting performance. In order to better understand these characteristics of the network between McGill and NYU, we conducted numerous

measurements of throughput, round trip time, jitter, and the effect of packet size and burst length on throughput.

4.5.1 Packet Size

A previous study [61] demonstrated that packet size has a significant effect on LAN throughput, with larger packet size leading to greater received data rates. We expected to find similar results in an Internet environment, so measured the throughput achieved for various packet sizes transmitted at a fixed rate between McGill and NYU. When the transmission rate was low (1.5 Mbps and 13 Mbps), packet size had little effect, but interestingly, at high transmission rates (>20Mbps), we were able to achieve maximum throughput with a *medium* sized packet of approximately 6000 bytes. Based on these results, we opted to use a packet size of 1500 bytes for the AC-3 stream, and 6912 bytes for the uncompressed format, since the effect of packet size is not significant in these two cases.

4.5.2 Packet Loss

The pattern of packet loss is very important since it determines the loss recovery algorithm we should use. Despite contrasting experiences of researchers working with the high-speed Abilene backbone [62], our observations were consistent with previous results demonstrating that packet losses are typically bursty [63][64]. Figure 4.6 shows one of our packet loss measurements at 13Mbps, with a packet size is 6912 bytes over a 20 minute trial. This means that the transmission rate of the packet is 250 packets per second. While the overall packet loss rate is negligible (0.6%), the number of dropped

packets in some of these episodes is alarmingly high. Although, there are only 13 episodes in which the number of lost packets is larger than 5, they count 57% of the total packet loss. It is highly impractical to use a simple FEC to recover from this kind of packet loss.

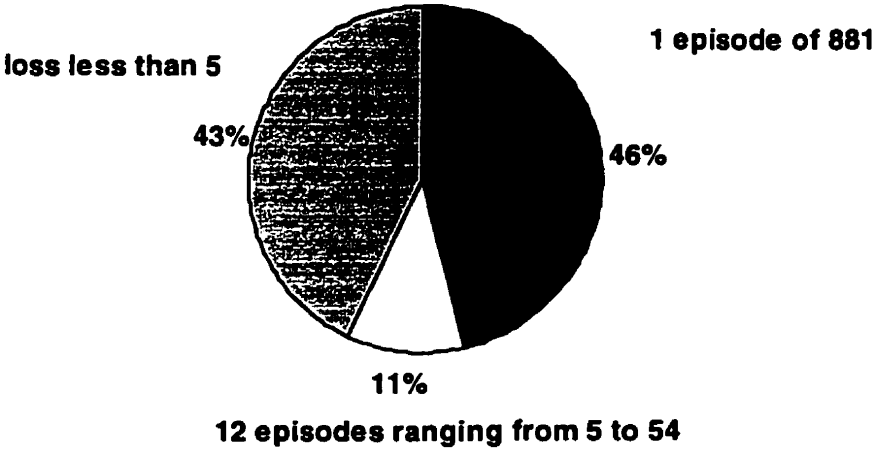


Figure 4.6. The bursty nature of packet loss. While there are only 13 bursts of packet loss that could not be recovered using an FEC scheme, these bursts accounted for 57% of the total packet loss. In particular, a single large burst, such as the one shown here of 881 packets, is not atypical for busy periods of the day.

Under conditions of packet loss caused by network congestion, it is likely that the congestion is short-lived, in which case, retransmitted data can arrive without difficulty. However, it is also possible that the retransmission will face similar or worse congestion than that which caused the original packet loss, especially if the resend occurs in close proximity to the initial transmission. By introducing a receive buffer that is longer than the expected burst duration, retransmission congestion problems can be reduced

significantly. The remainder of this section explores our design decisions concerning the management of packet loss as well as the experiments conducted to validate the design.

4.5.3 AC-3 retransmission

In earlier sections, we suggested that a TCP based retransmission might be better suited for AC-3 data than a simple UDP based scheme. In order to verify this hypothesis, we carried out simulations using both TCP and simple[1] UDP retransmissions to recover from packet loss caused by manually generated network congestion. Figure 4.7 summarizes the results of these tests, comparing TCP and UDP retransmission protocols under varying levels of network congestion. The generated congestion has a ten-second duration while the receiver allows a maximum of five seconds delay before an audio packet must be played.

Under conditions of mild network congestion, TCP demonstrates a 100% recovery rate, while UDP, due to its unreliable nature, achieves slightly less. Thus, without a considerable amount of hand-tuning of the UDP recovery mechanism, TCP is preferable under modest network loads. This should come as no surprise, since TCP has been optimized for such environments.

However, as network congestion increases, TCP does not cope as well. In fairness, TCP was designed to be *well behaved* to the network under such conditions, whereas our needs are somewhat greedy. As can be seen in Figure 4.7, for our particular experimental

configuration, UDP is better able to recover from network congestion brought on by competing traffic in excess of 30 Mbps. Although TCP will eventually deliver 100% of the packets, they do not arrive in time for playback. Thus, the effective recovery rate of TCP is significantly lower than our UDP protocol.

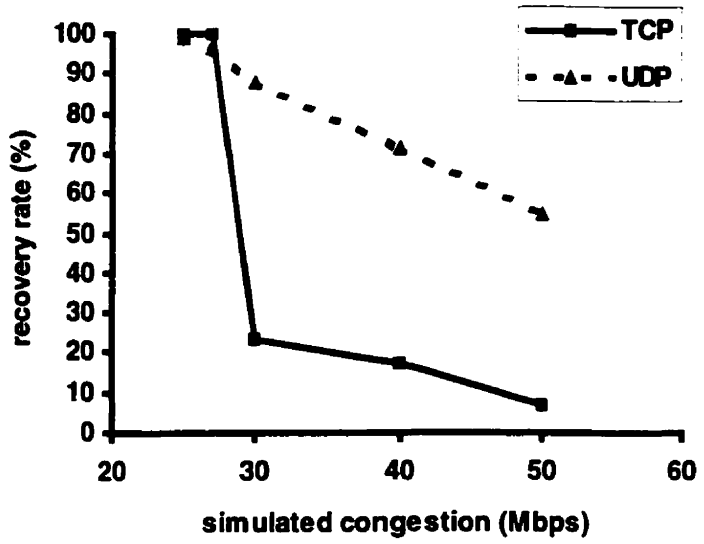


Figure 4.7. UDP- vs. TCP-based loss recovery protocols under network congestion. While competing throughput is under 30 Mbps, TCP is able to recover fully, but once congestion increases beyond this point, UDP is preferable for our application.

Interestingly, we note that despite the quantitative superiority of UDP (70% recovery vs. 17% for TCP) under severe congestion, there is little observable difference in terms of the audio quality. To a human listener, the resulting gaps in the audio stream are easily discernable, regardless of whether that gap lasts for 80 ms or 800 ms. In fact, it may be

[1] For a simple UDP retransmission scheme, the lost data packets are retransmitted only once.

preferable to mute for several seconds rather than play a segment corrupted by missing packets.

4.5.4 Layered Coding Congestion Control for Uncompressed Audio

In order to evaluate the effectiveness of the layered coding congestion control mechanism for uncompressed audio, we tested its performance under conditions of varying network congestion.[1] Again, we produced congestion manually, by pumping a dummy stream of up to 17 Mbps into the network at the source, in parallel with our 13 Mbps audio stream,

for a duration of approximately 30 seconds. Figure 4.8 presents the results of lost packet recovery both with and without layered coding congestion control. While the program is unable to recover all lost packets in time for playback under conditions of severe congestion, it performs significantly better with the layered coding congestion control mechanism than without it. From the listener's perspective, there is a region at the left of the graph in which the layered coding congestion control is able to achieve an unbroken playback stream, with occasional reduction in sample resolution, whereas a simple retransmission strategy results in occasional breaks in the music.

[1] Note that the experiments reported here were performed under different network conditions from those of the previous section. Whereas the data for Figure 5 was produced shortly after the first AES demo, with a lightly loaded receiver connected directly to the NYU backbone, the data for Figures 8 and 9 was obtained more recently, with the receiver running on a public machine several hops away from the backbone. This accounts for the discrepancy in additional data required to produce the same levels of congestion in the various experiments.

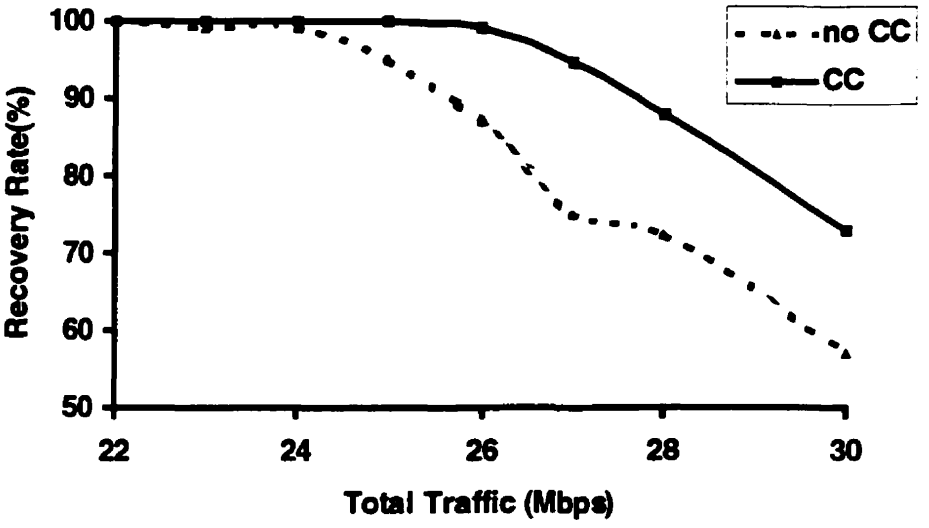


Figure 4.8. Demonstration of the effectiveness of lost packet recovery using layered coding congestion control (CC). The x-axis represents the sum of the bandwidth consumed by simulated congestion stream and the uncompressed audio stream. The y-axis indicates the percentage of packets that are recovered by the protocol in time for playback.

Of course, the primary motivation for employing a congestion control mechanism is its damping effect on bandwidth utilization during periods of heavy network traffic. Although the positive feedback loop still exists, its gain is reduced. This effect is demonstrated in Figure 4.9, representing the rate of packet loss, which is a reasonable indication of network congestion, as a function of time. One can see that without congestion control, packet loss increases rapidly and remains above 50% for almost half of the congestion period. From our experience, timely recovery becomes exceedingly difficult once the loss rate exceeds this level. However, with congestion control, the overall level of packet loss is lower, as is the rate of its increase during congestion. As a result, the program is able to recover 88% of the audio packets, as opposed to 77% without congestion control.

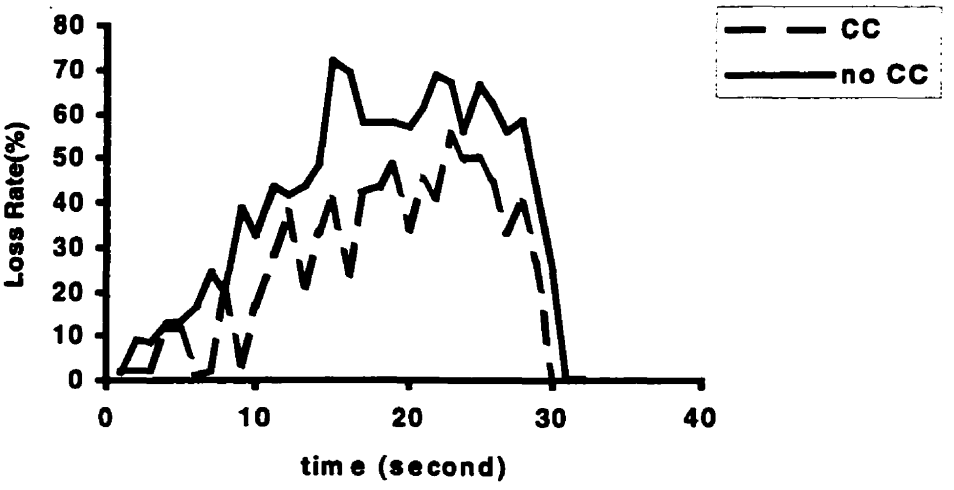


Figure 4.9. The effect of layered coding congestion control (CC) in relieving network load, under an artificially generated congestion stream of 30 seconds.

4.6 Conclusions

Before discussing the lessons learned from these experiments and demonstrations, it is helpful to distinguish between the requirements for our one-way audio transmission (e.g. broadcasting) and those of a two-way interactive audio transmission (e.g. teleconferencing). While the former are generally tolerant of several seconds of delay, incurred by buffering, this is not the case for interactive applications, such as remote music teaching, where the maximum tolerable latency is very low. For example, studies by telephony companies suggest that human conversation cannot tolerate latency in

excess of 200ms. For musical interaction this number is considered to be significantly lower.

4.6.1 Effective Loss Recovery

Although different applications require completely different architectures, the fundamental problem of audio over Internet is flow control, that is, efficient transmission and reliable recovery of lost data. For high bandwidth, unidirectional music streaming, we found automatic repeat request (ARQ) to be the most effective loss recovery mechanism, allowing for perfect playback of the audio stream, provided network congestion remained moderate. Unlike forward error correction (FEC), which incurs the overhead of additional bits on every transmission, ARQ need only increase its sending rate to recover from lost data. Not only is this approach more appropriate under conditions of bursty packet loss, it is also helpful in minimizing bandwidth requirements, which, in the case of uncompressed audio, are already very high.

As a counterpoint, in a recent demonstration of HDTV over Internet[1] [65], researchers at the University of Washington successfully made use of FEC for a 40 Mbps compressed transmission. However, their network path afforded two orders of magnitude more bandwidth, with their host-to-network connectivity operating on Gigabit (vs. 100 Mbit) Ethernet. Obviously, these differences do not permit an apples-to-apples comparison. While FEC proved effective for the HDTV trial, it could easily lead to congestion problems when spare bandwidth is limited.

[1] Please see www.washington.edu/researchtv/special/sc99.html

In general, one must make a trade-off between audio quality and latency. For interactive audio applications with a hard upper limit on latency, it is difficult to implement a perfect loss recovery protocol. If total delay is to be kept only marginally greater than network latency, it is nearly impossible to use an ARQ implementation since any retransmission request and resend involves a minimum of one round-trip delay. In this case, one would have no choice but to employ FEC. For example, in remote music teaching, if the participants are able to tolerate occasional breaks in the music, perhaps by repeating the broken session, then FEC should be employed to minimize latency and improve interactivity. In fact, we are in the process of implementing an FEC recovery protocol for AC-3 applications requiring low latency, since the lower bandwidth demands of AC-3 are unlikely to create congestion problems.

4.6.2 TCP as Transmission or Retransmission Protocol

Although, TCP is not an ideal protocol for bulk data transmission, various modifications can be made to improve its throughput [55]. For example, the size of the sliding window advertised by the receiver limits the amount of data that the sender can transmit pending an explicit acknowledgement [56]. To compensate, we increase the TCP buffer size on both sender and receiver, thereby improving throughput [57]. The selective acknowledgement protocol, along with various other extensions [66][67], is designed to improve the performance of TCP in the event of multiple missing segments within a window. Although, these modifications cannot change the fact that TCP, over a lossy network, is limited by its congestion control algorithm, we believe that a properly tuned

TCP based packet recovery protocol could recover most of the packet loss experienced by an AC-3 stream. However, this would likely not be feasible for high bandwidth uncompressed audio, nor interactive music applications in which minimal latency is imperative.

Our experiments further indicated that vanilla TCP is a good candidate as the retransmission protocol for AC-3 data, provided packet loss is not severe. In this case, TCP is only used for relatively small amounts of data communication, so delays related to its robust algorithms tend to be of minimal consequence. Exhaustive testing, in which the system ran uninterrupted for several days, demonstrated the robustness of this approach.

Based on the strengths of TCP, we are developing a loss recovery protocol on top of UDP. This would provide many of the desirable features of TCP, in particular, flow control and reliability, while offering a less restrictive congestion avoidance mechanism. For example, to maximize performance, the slow start algorithm [56] will not be implemented. However, some congestion control is still required, in particular under severe packet loss, as best-effort (i.e. greedy) delivery over the Internet is highly problematic [68].

4.6.3 Multicasting

Although our experiments concentrated on one-to-one communication, there are likely far more applications for this work that involve simultaneous transmission to multiple

recipients. Parallel unicasting to multiple hosts, as is the practice of many network-based applications, simply does not scale, and hence, multicasting is required [61]. Fortunately, the algorithms we have employed lend themselves easily to a multicast adaptation, and we will be exploring this direction in the near future.

4.6.4 Coping with Congestion

An important determinant in the evaluation of a software system is its performance under extreme conditions, that is, stress testing. In our application, this is related to the system's performance under conditions of severe congestion. While no system can assure 100% data transfer in this case, we can relieve congestion by employing the appropriate congestion control algorithms. We have already demonstrated that a system with proper congestion control suffers less packet loss during a severe congestion than a system without congestion control.

Theoretical and experimental study has shown that a single traffic stream with no congestion control can cause congestion collapse in a busy network, disastrous to all users. Our results are consistent with these studies. In response, the network community is promoting new routing algorithms that punish those traffic streams that do not employ congestion control mechanisms.

The design and implementation of a congestion control algorithm for end-to-end real time media communication is an active research area. While the congestion control utilized by TCP is mature and proven, it is not suitable for real time applications. Instead, most

ongoing studies follow the approach of layered coding. Unfortunately, a general purpose solution is impossible, as different media such as raw video, linear PCM audio, and AC-3 audio, each have different characteristics that may or may not lend themselves to any one layered implementation. For example, our simple implementation of layered coding appears to be effective for uncompressed audio transmission , but cannot be applied to AC-3.

As a final note on this topic, if the bandwidth of an audio stream is high relative to the available capacity, slight reductions achieved through layered coding will likely prove less effective than TCP congestion control. From our experience, unless layered coding can reduce bandwidth substantially without a serious decrease in observed quality, a better solution for congestion control may be to cut the transmission entirely for a certain period.

4.7 Real-world Demonstration

Although considerable effort went into the design and development of the system, we must confess to being surprised by how well it worked. Given the unreliability of network communications and the unavailability of bandwidth reservations, at least a few glitches were expected. Indeed, during development and tuning, various problems with the network configuration were noted, each of which could have proved fatal to the demonstration.

Following a cautious initial demonstration employing a twenty second buffer to the Cantor Film Center, the delay was reduced to a mere three seconds. When this failed to introduce interruptions to the audio stream, the network news feed into NYU was resumed, thereby generating considerable competing traffic at the receiver end. At this point, several interruptions to the Cisco IP/TV video stream (requiring similar bandwidth to our audio system and utilizing the same delay) were observed. However, at no point during the demonstration did the audio break from a continuous stream.

Chapter 5: Conclusions and Possible Improvements

This thesis described the effort of building an audiovisual communication system for the SRE. Chapter 1 introduced the rationale for building the SRE. Chapter 2 reviewed the technical issues related to the development of the testbed. Chapters 3 and 4 introduced the implementation of two systems that support audiovisual communication in the SRE. The first enabled video and audio communication between the initial two test rooms, located on the same LAN at CIM. The second was for streaming high-fidelity, multi-channel audio over the Internet. Although both systems proved successful, they still require some improvements in order to support all of the targeted SRE applications, the most important being the reduction of latency. This is limited by the available network architecture and transmission protocols. In Chapter 3, the possibility of reducing latency by constructing a hybrid system that transmits both raw and compressed video was explored. Finally, this chapter discusses several alternatives that might help to improve the performance of the system.

5.1 Network Architecture Improvement

The ideal solution is to provide a network with sufficient bandwidth so that latency can be reduced to the propagation delay along the network. For the initial SRE testbed, in which the rooms are located either on the same LAN (CIM) or on the same campus (McGill), it is quite feasible to build such a high performance network. For example, we

can replace the 100 BaseT Ethernet connections with a Gigabit, or even 10-Gigabit connection. ATM is also a possible candidate, although a recent trend is to send IP directly over the optical connection, as is done by CAnet 3.

Unfortunately, if the remote location is more distant, such as another city, it is not trivial at present to provide connections with such high bandwidth. Even though the backbones of several high performance networks (e.g. Internet 2 and CAnet 3) carry this bandwidth, it is very difficult to lay an end-to-end connection to an arbitrary desktop. At Supercomputer 99 (SC99), streams of data at 10Gbps were delivered from Microsoft Corporation and the University of Washington at Seattle to the SS99 exhibition hall in Portland. While, this was only for demonstration purposes, it does indicate that such high bandwidth transmission capabilities are only a short time away.

5.2 Forward Error Correction

In Chapter 4, a system that is able to stream high quality, multi-channel audio over the Internet was developed. Since the transmission was unidirectional, the system could afford to have a latency of a few seconds, allowing us to use ARQ as the error correction scheme. However, as stated in Chapter 2, such latency is unacceptable for interactive applications. Therefore, FEC is the better-suited alternative.

Unfortunately, as has been shown in Chapter 2, FEC's ability to recover is strictly dependent on the loss characteristics of the underlying network, and cannot cope with

large bursts of packet loss. Our observations concerning the connection between McGill and NYU demonstrate that large bursts of packet loss do occur even in high performance networks. Therefore, FEC's benefit of reducing latency is offset by its inability to recover from all packet losses.

In Chapter 4, attempts were made to transmit two kinds of audio data: AC-3 and uncompressed audio. This section concentrates on AC-3, which exhibits less severe loss characteristics due to relatively low bandwidth requirements. In contrast, the packet losses resulting from the transmission of uncompressed audio can be quite serious, often proving unmanageable for a FEC recovery scheme. However, the same principle of FEC can be applied to uncompressed audio if the underlying network is improved in the future.

Traditional FEC involves exclusive-OR operations [31]. The idea is to send a redundant packet, every n+1st packet, obtained by exclusive-ORing the other n packets. This mechanism can recover from a single loss in an n packet message. It is a very simple mechanism, although it increases the send rate of the source by a factor of 1/n, and adds latency since n packets have to be received before the lost packet can be reconstructed. Recently, several other FEC mechanisms that are designed for transmitting audio and video over the Internet have been proposed [33,34,35,36,37]. Most of them also involve the idea of layered coding, which was described in Chapters 3 and 4. Based on this previous research, an FEC scheme for the AC-3 audio stream is now proposed.

As introduced in Chapter 2, the default rate of AC-3 is 384Kbps. However, it is possible to generate AC-3 with either higher or lower bit rates in order to obtain different audio quality. For example, the Dolby Encoder DP569 is able to generate an AC-3 stream at 640kps, 320kps and 256kps, 128kps. In Chapter 4, it was mentioned that AC-3 data is typically embedded in an AES/EBU stream and it is possible to extract the AC-3 stream; for the following discussion, we assume that the raw AC-3 data has already been extracted.

Figure 5.1 demonstrates our proposed FEC scheme for AC-3. The audio source is encoded into multiple AC-3 streams at different bit rates, stored in separate audio buffers. The packet-send engine is a software module that generates AC-3 data packets. For each data packet *n*, its payload field is divided into two parts. The first stores the AC-3 data at its highest bit rate (640kps) for the *n*th data packet, while the second stores the AC-3 data of packets n-m to n-1 at a lower bit rate (e.g. 320 kps). If data is lost during transmission, the receiving system will wait for the next received packet and replace the lost packets with the redundant component from the newly arrived packet. Such a system is able to recover from a packet loss involving up to m-1 packets.

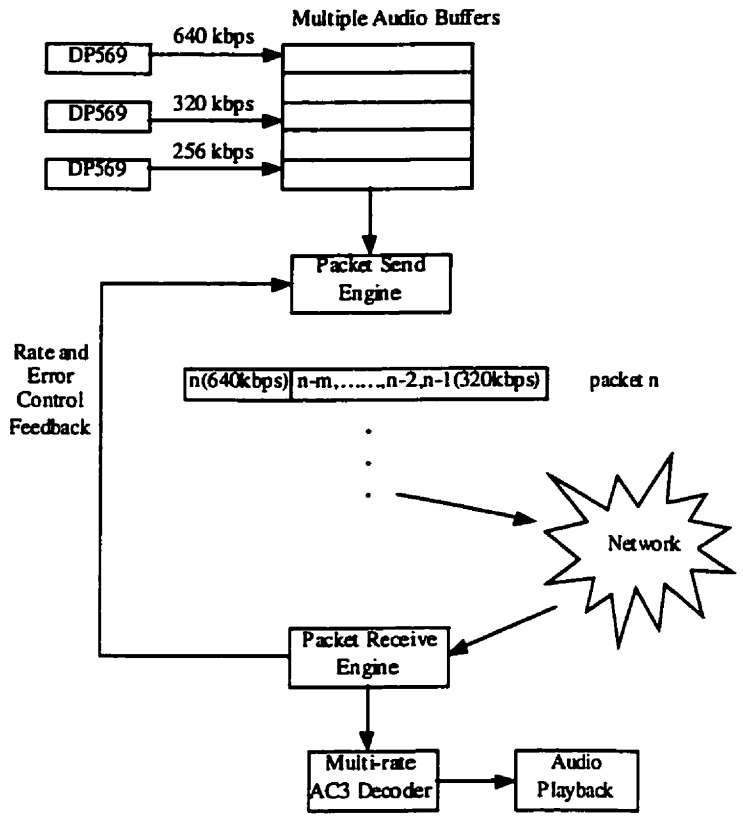


Figure 5.1. AC-3 transmission with FEC

However, one should realize that this design has several limitations. Since the value of m determines the maximum recoverable packet loss, any greater loss cannot be sustained without error. While this might suggest using a larger value of m, increasing this value also increase the consumption of bandwidth, which in turn increases the possibility of further packet loss. Besides the value of m, the bit rates for the redundant data can also be changed as long as they are lower than the default bit rate. In general, the relation of cost (overhead bandwidth introduced by redundant data) and reward (maximum packet recovery) has to be evaluated carefully in order to obtain the optimal solution.

The usage of AC-3 with multiple bit rates requires the AC-3 decoder at the receiving end to have the ability of dynamically decoding AC-3 data with various bit rates in real time. While the Dolby Decoder DP562 does not support this function, we believe that it would not be difficult to develop a software AC-3 decoder to do so as the decompression of AC-3 does not require a lot of CPU power.

The packet-receive engine in Figure 5.1 sends the control information, such as the degree of network congestion and the recovery rate under the current FEC settings, back to the packet-send engine. Based on this feedback, the packet-send engine is able to change the FEC settings dynamically in order to obtain the best cost-reward balance. For example, if the observed packet loss is small, each packet can carry less redundant data. On the other hand, if network congestion leads to large packet losses, we can increase the value of m while decreasing the bit rate of the redundant data, or perhaps, the bit rate of the original data.

The previous sections describe a possible design of an audio transmission system that has very low latency while still allowing for efficient recovering from packet loss. Although it is designed for an AC-3 audio stream, the same principle can easily be applied to other media such as uncompressed audio or various video formats. Of course, the implementation is strictly dependent on the type of data being transmitted and the characteristics of the underlying network.

References:

1. M. Weiser, Some computer science problems in ubiquitous computing. Communications of the ACM, Vol. 36, No.7, July 1993.

2. M. Coen, Design principles for intelligent environment. Proc. of 1998 AAAI Spring Symposium on Intelligent Environments, Standord, CA, March 1998.

3. J. Cooperstock, S. Fels, W. Buxton and K. Smith, Reactive environments: Throwing away your keyboard and mouse. Communications of the ACM, Vol. 40, Num. 9, 1997

4. www.cim.mcgill.ca/~jer

5. R.Rasch, Synchronization in performed ensemble music. Acustica, 43, 121-131,1979

6. J. Day and H. Zimmermann, The OSI reference model. Proc. of the IEEE, Vol.25, Oct. 1995

7. D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. Proc. of SIGCOMM'90, Philadelphia, PA, 1990

8. S. Floyd, V. Jacobson, S. McCanne, C. Liu, L. Zhang. A reliable multicast framework for lightweight sessions and application level framing. Proc.of SIGCOMM'95, Boston, MA, Sep. 1995

9. S.McCanne, V. Jacobson. vic: A flexible framework for packet video. ACM Multimedia, San Francisco, CA, Nov. 1995.

10. R. Frederick. Experiences with real-time software video compression. Proc. of the Sixth International Workshop on Packet Video. Portland, OR, Sep. 1994

11. T. Turletti and C. Huitema. Videoconferencing on the Internet. IEEE/ACM Transactions on Networking, Vol. No.3, June, 1996

12. M. Willebeek-LeMair, Z. Shae. Videoconferencing over packet-based networks. IEEE. Journal on selected areas in communications. Vol. 15. No. 6, Aug. 1997

13. M. Baldi, and Y. Ofek, End-to-end delay of videoconferencing over packet switched networks. 9th IEEE Workshop on Local and Metropolitan Area Networks. Banff, Alberta, Canada, May 1998.

14. J. Dixon, A comparison of audio and video synchronization rates (Quarterly Technical Report). New York: Society of Motion Picture and Television Engineers, 1987.

15. W. Pennebaker and J. Mitchell. JPEG still image data compression standard. Van Nostrand Reinhold, New York, 1993

16. www.ijg.org

17. J.Mitchell, D.Gall, C. Fogg. MPEG video compression standard. Chapman and Hall, 1996

18. B.Haskell, A.Puri, A. Netravali. Digital video compression standard: An introduction to MPEG2. Chapman and Hall,1996

19. L. Chiariglione. MPEG and multimedia communications. IEEE. Transactions on Circuits and Systems for Video Technology, Vol. 7, Num.1, Feb. 1997

20. B. Girod, E. Steinbach, N. Farber. Comparison of the H.263 and H.261 video compression standards. First International Symposium on Photonics Technologies and Systems for Voice, Video and Data communication, Oct. 1995

21. ATSC A/52. Digital audio compression (AC-3) standard. United States Advanced Television Systems Committee (http://www.atsc.org/Standards/A52).

22. J. Postel. Transmission control protocol. RFC 793, USC/Information Sciences Institute, Sep. 1981.

23. J. Postel. User datagram protocol. RFC 768, USC/Information Sciences Institute, Aug. 1980.

24. V. Jacobson. Congestion avoidance and control. Proceedings of SIGCOMM'88, 1988.

25. M. Allman, V. Paxson, W. Stevens. TCP congestion control. RFC 2581, USC/Information Sciences Institute, April, 1999.

26. V. Jacobson. Modified TCP congestion avoidance algorithm. Technical report, Apr. 1990, URL: ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

27. K. Fall, and S. Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. Computer Communication Review, Vol.26, No. 3, July 1996.

28. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A transport protocol for real-time applications. RFC 1889, USC/Information Sciences Institute, Aug.,1996.

29. S. Lin and Costello. Error control coding: Fundamentals and applications. Prentice-Hall, Englewood Cliffs, NJ, 1983

30. R. Blahut. Theory and practice of error control codes. Addison-Wesley, Reading MA, 1983.

31. N.Shacham, P. McKenney. Packet recovery in high-speed networks using coding and buffer management. Proc. IEEE INFOCOM, Vol.1, San Francisco, CA, June,1990.

32. C. Huitema. The case for packet level FEC. Proc. IFIP 5th International workshop on Protocol for high speed networks, INRIA, Sophia Antipolis, France, Oct. 1996

33. J. Nonnenmacher, E. Niersack. Reliable multicast: Where to use FEC. Proc. IFIP 5th International workshop on Protocol for high speed networks, INRIA, Sophia Antipolis, France, Oct. 1996

34. J. Nonnenmacher, E. Niersack and D. Towsley. Parity-based loss recovery for reliable multicast transmission. Proc. of SIGCOMM'97, Cannes, France, Sep. 1997

35. J. Bolot, A. Veg-Garcia. Control mechanisms for packet audio in the Internet, Proc. IEEE. INFOCOM, San Francisco, CA, April 1996.

36. J. Bolot, S. Fosse-Parisis, D. Towsley. Adaptive FEC-based error control for Internet Telephony. Proc. IEEE INFOCOM, New York, March 1999.

37. P. Lee, Forward error correction coding for packet loss protection. First International Packet Video Workshop, Columbia University, New York, May 1987

38. S. Pejhan, M. Schwartz, D. Anastassiou. Error control using retransmission schemes in multicast transport protocols for real-time media. IEEE/ACM Transactions on Networking, Vol.4, No.3, June 1996.

39. S. Jacobs and A. Eleftheriadis. Streaming video using dynamic rate shaping and TCP flow control,

40. D. Sisalem, F. Emanuel, H. Schulzrinne. The direct adjustment algorithm: a TCP friendly adaptation scheme. Preprint.

41. D. Sisalem, H. Schulzrinne. The loss-delay based adjustment algorithm: a TCP friendly adaptation scheme. Proc. International Workshop on Network and Operating System Support for Digital Audio and Video. Cambridge, England, Jul. 1998.

42. L. Vicisano, J. Crowcroft, L. Rizzo, TCP-like congestion control for layered multicast data transfer. Procc IEEE INFOCOM'98, San Francisco, CA, March 1998.

43. T. Bailly. A technique for adaptive voice flow control in integrated packet networks, IEEE Transaction. Communication, Vol. COM-28, Mar. 1980.

44. G. Karlsson and M. Vetterli. Subband coding of video signals for packet-switched networks, Proc. SPIE Conf. Visual Commun. Image Processing II, Vol. 845, Cambridge, MA. Oct. 1987.

45. G. Karlsson and M. Vetterli. Three dimensional sub-band coding of video, Proc. ICASSP'88, New York, April 1988

46. G. Karlsson and M. Vetterli. Subband coding of video for packet network, Optical Engineering, Vol. 27, July 1988.

47. M.Vetterli. Multi-dimensional sub-band coding: some theory and algorithms. Signal Processing, Vol. 6, Feb. 1984

48. H. Musmann. Advances in picture coding, Proc. IEEE. Vol. 73, April 1985

49. N. Shacham. Multipoint communication by hierarchically encoded data. Proc. IEEE. INFOCOM, Vol.1, Florence, Italy, May 1992.

50. W. Zhao, M. Willebeek-LeMair, P. Tiwari. Efficient adaptive media scaling and streaming of layered multimedia in heterogeneous environment. Proc. International Conference on Multimedia Computing and Systems. Florence, Italy, 1999

51. P. Nee, K. Jeffay and G. Danneels, The performance of two-dimensional media scaling for Internet Videoconferencing. Proc. IEEE International Workshop on Network and Operating System Support for Digital Audio and Video. St. Louis, MO. May 1997

52. D. Taubman, A. Zakhor. Rate and Resolution Scalable 3D Subband Coding of Video. Proc. of SPIE Symposium on Digital Video Compression on Personal Computers: Algorithms and Technologies, San Jose, CA, Feb. 1994.

53. J. Cooperstock, K. Tanikoshi, G. Beirne, T. Narine, and W. Buxton. Evolution of a Reactive Environment. Proc. of CHI'95, Conference on Human Factors in Computing Systems, Denver, May 1995.

54. IEEE Journal on Selected Areas in Communications, Vol. 14, Jan. 1996

55. R. Cohen, S. Ramanathan. TCP for high performance in hybrid fiber coaxial broad-band access networks. IEEE/ACM Transactions on Networking, Vol.6, No.1, Feb. 1998.

56. R. Stevens. TCP/IP illustrated. Vol. 1, Addison-Wesley, Reading, Mass. 1994.

57. J.C. Mogul. IP network performance, in Internet System Handbook, eds, D.C. Lynch and M.T.Rose, Addison-Wesley, Reading , Mass., 1993.

58. M. Coen and K. Wilson, Learning spatial event models from multiple-camera perspectives, Industrial Electronics Society, IECON '99 Proceedings, Vol.1, 1999.

59. S. Arseneau and J.R. Cooperstock. Real-time image segmentation for action recognition. Proc. of IEEE PACRIM, Pacific Rim Conference on Communications, Computers, Visualization and Signal Processing, Victoria, August 1999.

60. AES White Paper. Technology Report TC-NAS 98/1: Networking audio and music using Internet2 and next-generation Internet capabilities. http://www.aes.org.

61. J. R. Cooperstock and S. Kotospoulos. Why use a fishing line when you have a net? An adaptive multicast data distribution protocol." Proc. of USENIX '96. San Diego, CA, 1996.

62. D. Richardson. Personal Communications. Dec. 20, 1999.

63. M. Borella, D. Swider, S. Uludag, G. Brewster. Internet packet loss: measurement and implications for end-to-end QoS. Proceedings of the 1998 ICPP Workshops on Architectural and OS Support for Multimedia Applications/Flexible Communication Systems/Wireless Networks and Mobile Computing, 1998.

64. V. Paxson. End-to-end Internet packet dynamics. IEEE/ACM Transactions on Networking. Vol.7, No.3, June 1999.

65. www.washington.edu/researchtv/special/sc99.html.

66. M. Mathis, J. Mahdavi, S. Floyd, A. Romanow. TCP selective acknowledgement options. RFC 2018, USC/Information Sciences Institute, Oct. 1996.

67. M. Mathis, J. Mahdavi. Forward acknowledgement: refining TCP congestion control. Proceedings of SIGCOMM 96. August 1996.

68. S. Floyd. Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking, Vol. 7, No. 4, Aug. 1999.