Transport and Display Mechanisms For Multimedia Conferencing Across Packet-Switched Networks[*]

K. Jeffay, D.L. Stone, F.D. Smith

University of North Carolina at Chapel Hill
Department of Computer Science
Chapel Hill, NC 27599-3175 USA
{jeffay,stone,smithfd}@cs.unc.edu

September, 1993

Abstract: A transport protocol that supports real-time communication of audio/video frames across campus-area packet switched networks is presented. It is a "best effort" protocol that attempts to ameliorate the effect of jitter, load variation, and packet loss, to provide low latency, synchronized audio and video communications. This goal is realized through four transport and display mechanisms, and a real-time implementation of these mechanisms that integrates operating system services (*e.g.*, scheduling and resource allocation, and device management) with network communication services (*e.g.*, transport protocols), and with application code (*e.g.*, display routines). The four mechanisms are: a facility for varying the synchronization between audio and video to achieve continuous audio in the face of jitter, a network congestion monitoring mechanism that is used to control audio/video latency, a queueing mechanism at the sender that is used to maximize frame throughput without unnecessarily increasing latency, and a forward error correction mechanism for transmitting audio frames multiple times to ameliorate the effects of packet loss in the network. The effectiveness of these techniques is demonstrated by measuring the performance of the protocol when transmitting audio and video across congested networks.

Published in: *Computer Networks and ISDN Systems*, Vol. 26, No. 10, (July 1994) pp. 1281-1304.

[*] This work supported in parts by grants from the National Science Foundation (numbers CCR-9110938 and ICI-9015443), and by the Digital Equipment Corporation and the IBM Corporation.

1. Introduction

Inexpensive hardware for processing digitized audio and video data is rapidly becoming available for desktop personal computers. At the same time, high network bandwidth at relatively low price is widely available at the desktop. These developments have stimulated interest in the application of computer-based conferencing to support more effective collaboration among teams of workers in various fields. Computer-based conferencing systems attempt to create a feeling of “virtual presence” among team members not in the same physical location by providing shared access to computer-based materials along with links for human communications (*e.g.*, [8, 9, 14, 20]). For example, a team of software designers might use a shared “whiteboard” (drawing) application to sketch the logical structure of a program, a shared editor to outline the documentation, and shared audio and video links to discuss alternatives and reach consensus (see Figure 1.1). In the following, we focus on the problem of support for communication of the audio and video portions of a computer-based conference.

To realize the full potential of collaboration-support systems, one must combine the convenience of using the network and computer at one’s own desk with the power and quality of expensive, specialized videoconferencing equipment. This means that new capabilities for processing digital audio and video are required for desktop computing environments, especially in operating system and network services. The problem of providing operating system support for these applications has been addressed by ourselves and others [3, 6, 11, 17]. The work described here assumes appropriate services, especially those for real-time scheduling of tasks, are available in the underlying operating system. We focus instead on the network services and those aspects of the conferencing application necessary to support human-to-human communication using digital audio and video.

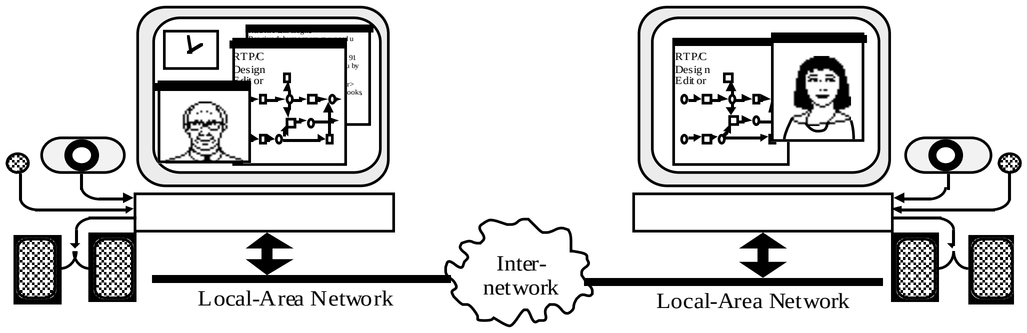


Figure 1.1: A multimedia conferencing system.

There are a large number of research and commercial product development activities that seek to provide solutions for computer-based conferencing. Many of these involve application of specialized technology developed for video teleconferencing and standardized in the H.320 series of CCITT. These systems rely on ISDN or other facilities to establish dedicated point-to-point communications links. Considerably more interest has been focused on the potential of ATM-based networks to provide an integration of audio and video with other data and to provide services (*i.e.*, isochronous communications) that are well-matched to the requirements of audio and video.

In contrast, our research is concerned with how conferencing solutions can be obtained using (*a*) readily available commodity audio and video technology for workstations, and (*b*) todays' networks based on asynchronous communications (*e.g.*, ethernet, token rings, FDDI, T3, *etc.*) internetworked using bridges and routers. We believe this work is important for the following reasons.

1. We expect that networks consisting of a heterogeneous mix of conventional LANs, bridges, routers, and ATM switches will be widely used during the evolution of networks towards ATM. Our work presents a solution that is particularly applicable if, as we believe, conventional LANs are widely used in the "last mile" to the desktop.

2. We demonstrate empirically that it is possible to support most user requirements for conferences across small (*e.g.*, campus area) packet-switched networks without using special network services, even in the presence of congestion. Thus, the work establishes a baseline for measuring the benefits and costs of more specialized solutions.

The design one adopts for computer-based conferencing depends on numerous factors including the architecture of the audio/video subsystem, the encoding and compression technologies employed, the available network bandwidth, and the availability of special network services. Most widely available audio/video subsystems digitize and display video based on NTSC standards for color video and provide high-speed compression and decompression of the data. In the NTSC standards, motion perception is achieved by acquiring and displaying still images at the rate of 30 per second (each still image is displayed for approximately 33 ms.). Compression of these images is essential for transmission over the networks considered here (*e.g.* to reduce the required bandwidth for 256x240 resolution, color, full motion video from 45 megabits per second to near 1 megabit per second). We consider only those subsystems employing compression algorithms that operate on independent still images (*e.g.*, using the JPEG standard).

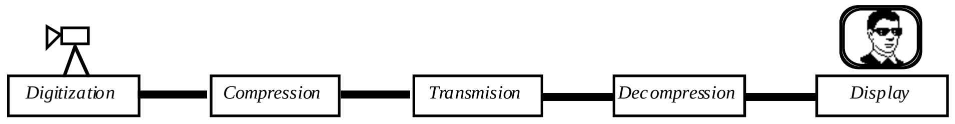


Figure 1.2: High-level system architecture.

The conferencing system considered in our work (see Figure 1.2) operates by acquiring and digitizing, at regular intervals, frames from a video camera and microphone attached to a participant's workstation. The resulting stream of frames is compressed and transmitted over the network to another participant's workstation where the frames are decompressed and displayed on attached monitors and speakers. (A frame is the logical data unit by which audio or video is acquired and displayed, normally corresponding to a single 33 ms image. A stream is a totally ordered, continuous sequence of frames.)

The work reported here does not consider the effect of special network services such as admission control and resource reservation [2]. While such services are clearly useful and important, they equally clearly imply additional costs to modify routing software at all network interconnection points, and possible reductions in the total available bandwidth.

The following section describes the conferencing system design by considering the users' requirements, the basic system structure, and the problems that are encountered in using today's networks. We follow in Section 3 with a description of the experimental system used to evaluate our solution along with an analysis of our metrics for the system. Section 4 describes the proposed transport and display mechanisms that are the central elements of this design. Section 5 presents the results of experiments using these mechanisms to transmit live digital audio and video over interconnected local-area networks and outline plans for future work. We conclude in Section 6 with a review of our results.

2. Conferencing System Design

2.1 User Requirements

Users will expect a computer-based conferencing system to provide audio comparable to a conventional telephone call and video that is a reasonable approximation to broadcast television (on a small-screen TV) in terms of color, image clarity, and motion perception. Some factors such as image resolution, color tones, and audio frequency response are directly determined by the particular equipment chosen and are not considered here.

Other important metrics that influence users' perceptions of a conference (and that can be controlled by the system software) are: the end-to-end latency of communications between the participants, the number of discontinuities (*i.e.*, frames that are either never played, or played multiple times), and the deviations from exact synchronization of audio and video. End-to-end latency is the elapsed time between the acquisition of a frame at the sender and the display of the frame at a receiver. The end-to-end latency should be kept under 250 ms for most conferencing applications [4].

A discontinuity occurs when frame $n + 1$ is not displayed immediately after frame n. This occurs if a frame is lost from the stream or does not arrive at the receiver in time to be played when the previous frame is finished. Because most communication in a conference is oral conversations, users demand much more of the audio than they demand of the video. Moreover, because there is less temporal coherence in the audio stream than in the corresponding video stream, users are quite sensitive to discontinuities in the audio. In contrast, users can tolerate significant discontinuities in the video stream. For example, if video frame n is not present at the receiver when it is time for it to be played, the preceding frame, frame $n - 1$, can be played in its place without adverse effect. Indeed if the late arrival of a frame is an isolated event, a user will not notice that a single frame has been replayed. By contrast, if a corresponding amount of audio data (*e.g.*, 33 ms worth) does not arrive on time, a replay of the previous 33 ms is easily noticed (as is playing nothing for 33 ms). Even with audio frames representing only 16.5 ms, discontinuities have a significant effect on users' perception of the conference.

Also related to users' perception of the conference is the synchronization between audio and video frames (*i.e.,* "lip sync"). For this work we assume that audio and video should never be more than 100 ms out of synchronization [18].

2.2 System Structure

The basic structure for computer-based conferencing systems using todays' commodity technologies is shown in Figure 2.1. It is useful to view the processing components as a multi-stage distributed pipeline through which frames flow from source to destination. In Figure 2.1, each processing stage is represented as a circle and storage buffers that hold individual frames are shown as squares. Under program control, each frame is digitized, compressed, transmitted over the network, received, decompressed, and displayed.

Specialized processors on audio/video hardware adapters are typically used to off-load computationally intensive compression and decompression from the system processor, as

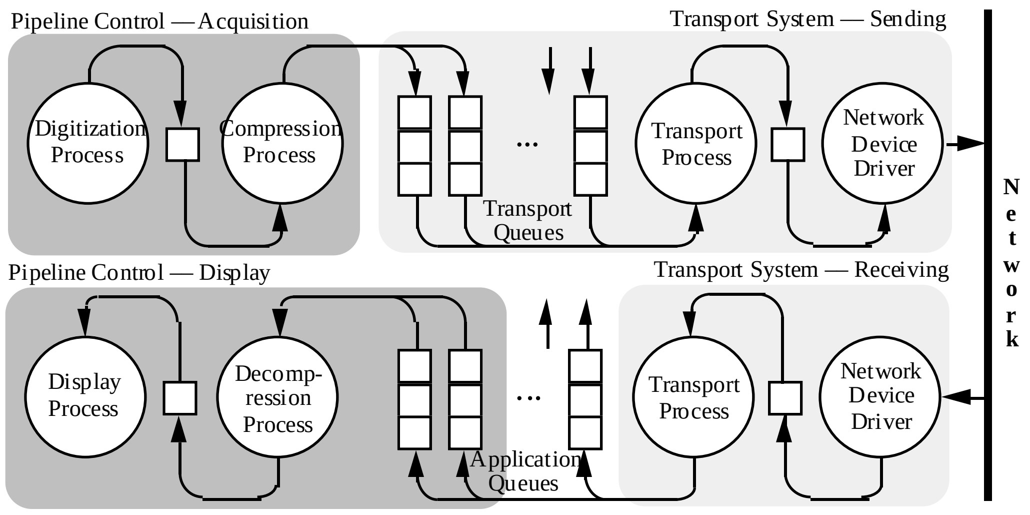


Figure 2.1: Transport system architecture.

well as to perform the digitization and display operations. Similarly, network adapters usually include specialized processors to handle medium access and transmission functions. The system CPU is, however, required to initiate and control these pipeline stages and, in some cases, move data between storage buffers on the adapters. The processing stages shown in Figure 2.1 are abstractions that represent elapsed times for major functions and combine work done by the CPU and specialized processors. Note that queues holding more than one frame exist only at the boundaries between the transport stages and the rest of the pipeline. Queues are not required at other stages because frames can be made to flow in real-time (given sufficient support for real-time resource allocation in the operating system). Since the network load is not amenable to such real-time control, transport send and receive queues are necessary.

On some networks (*e.g.*, 16 megabit token rings and FDDI), one network packet can hold an entire video frame and its corresponding audio. (At 256x240 resolution a compressed color video frame requires between 2,000 and 8,000 bytes of storage. Approximately 500 bytes are required for two channels of compressed audio sampled at very high rates.) On other networks (*e.g.*, ethernet), the frame must be segmented into multiple network packets for transmission. In some cases it is also possible to package all or part of the data from multiple frames into a single packet. The way frames are allocated to network packets has significant implications if the packets encounter problems in transmission as described in the next section.

2.3 Problems

In the network environment considered in our work, conferencing users cannot control how the network is used by other applications and by other workstations. The key difficulty to overcome in this environment is the effect of congestion in the network. Congestion manifests itself in two ways depending on the origin and nature of the congestion. First, as congestion in a local-area network increases, a workstation attached to that network encounters delays when attempting to access the shared medium to transmit data (*e.g.*, waiting for idle carrier in ethernet or a token on token ring). Second, as congestion increases on intermediate networks or at "store-and-forward" nodes such as bridges and routers, packets may encounter queueing delays. In case of severe congestion, packets may be lost at routers or bridges because buffer space is exhausted or the CPU is saturated. In either case, the packet delays and losses induced by congestion have serious impacts. Depending on the way frames are packaged into packets, the delay or loss of a single packet may result in discontinuities spanning one or more frames. Three specific congestion-related phenomena (jitter, load variations, and packet loss) are discussed further in the following.

Jitter

Jitter is the deviation in inter-frame arrival times induced by random variations in media access, and queueing delays at intermediate nodes. (Resource contention at the sending and receiving machines may also contribute to jitter.) For the systems considered here, a new frame from a stream must be available for display by the hardware at a precise rate (*i.e.*, one frame every 33 ms for NTSC video). Although these frames can be generated at the desired rate by the acquisition hardware, the exact rate at which frames arrive at a receiver can be grossly distorted by jitter. Jitter may cause discontinuities in the displayed stream (because a frame does not arrive in time) and hence has a direct impact on the users' perception of a conference.

Load variation

Related to the problem of jitter is that of dynamically changing network load. Methods for dealing with jitter, roughly speaking, accommodate short-term deviations in stream flows caused by short (millisecond) bursts of traffic on the network. It is also desirable to accommodate long-term deviations caused by gross changes in network traffic that persist for hundreds of milliseconds. When such changes result in long-term congestion, the throughput of frames between a transmitter and receiver may not be sufficient to sustain normal display rates. The effect is to increase both stream latency and the rate of discontinuities appearing in the displayed stream. The number of discontinuities increases because fewer frames are physically transmitted.

Packet loss

As the load on a network increases, packets may be lost as they pass through routers or bridges. Traditional communication protocols deal with packet loss through time-out and re-transmission mechanisms. These techniques cannot be used without introducing unacceptably high latency into the streams. Simply put, by the time it is realized that a frame is missing it is likely too late to avoid the impending discontinuity. If a frame has been segmented for transmission in multiple network packets, the loss of any one packet causes the entire frame to be lost. Similarly, if a lost packet contains all or part of a multi-frame audio sequence (*e.g.*, to compensate for long delays in media access time), then the effect on stream continuity is exaggerated. This is particularly disturbing to users because they have higher expectations for audio communications.

We focus here on transport-layer solutions to the problem of packet loss and do not consider methods such as approximating missing packets by interpolation. Interpolation techniques may increase the latency in the stream and require hardware-specific details of the compression and encoding technology employed. (Interpolation will, however, complement our solution.)

3. Experimental System and Metrics

We have constructed a testbed for experimenting with the transmission of live digital audio and video across IP networks. For acquisition and display we use IBM-Intel ActionMedia 750 hardware [7, 11] and IBM PS/2 workstations. The PS/2s run a real-time operating system we have developed (YARTOS [10]) and support UDP/IP. The network configuration used for experiments consists of 16 megabit token rings and 10 megabit ethernets, interconnected by bridges and routers (see Figure 3.1).

3.1 ActionMedia Hardware

In a system using ActionMedia 750 hardware, the CPU directs the acquisition, compression, and transmission of each frame including all movement of the data within the system as shown in Figure 3.2. The effective use of such hardware requires that real-time services be provided by the operating system [11].

The ActionMedia hardware provides a two stage pipeline for either acquisition or display of video data. The acquisition pipeline consists of digitization and compression stages;

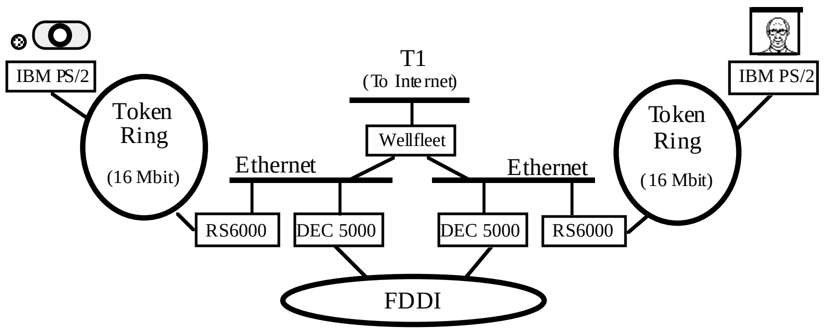


Figure 3.1: Experimental testbed.

the display pipeline consists of decompression and display stages. The acquisition and display stages each require 33 ms. The duration of the compression and decompression stages is a function of scene complexity and the algorithm employed. In our use of the system these stages typically require 20-25 ms. A compressed, color video frame (at 240x256 resolution) is between 6000-8000 bytes, yielding a data rate of approximately 1.7 megabits per second for full-motion video.

Audio frames are digitized and compressed in a single stage. The length of this stage is programmable. To reduce processing overhead, we typically use values of 16.5 or 33 ms. A 33 ms audio frame is a little over 500 bytes and the data rate for a 2-channel audio stream is approximately 128 kilobits per second.

As illustrated in Figure 3.2, there is a significant disparity between the lengths of the audio and video pipelines. For video frames, the acquisition and compression stages run sequentially. The earliest a video frame can be transmitted is approximately 60 ms after digitization begins (assuming no operating system or application overhead). In contrast,

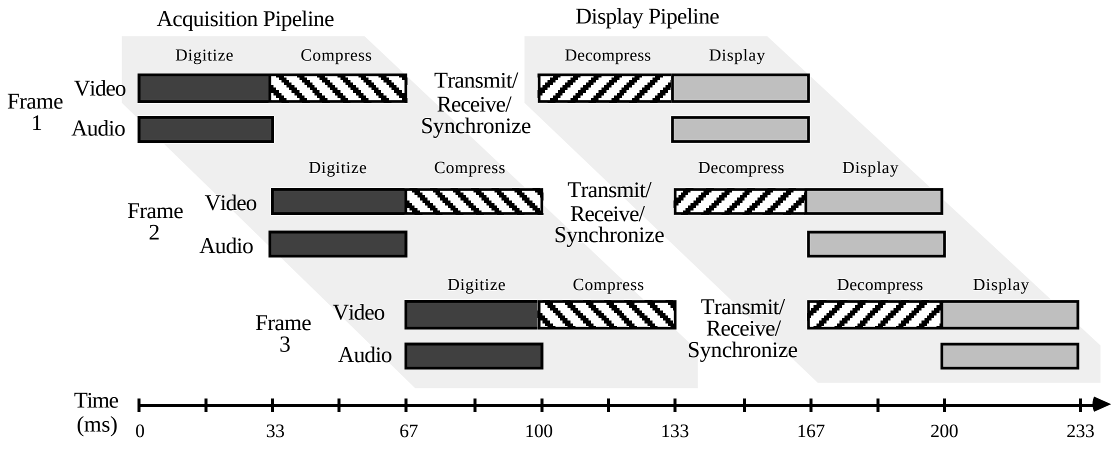


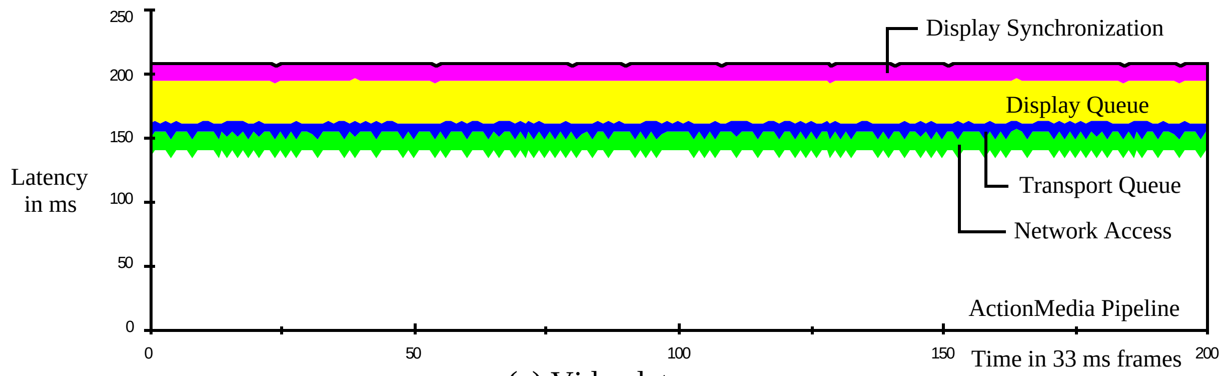
Figure 3.2: The ActionMedia audio and video pipeline for 33 ms frame times.

the audio data corresponding to a video frame can be transmitted as soon as the acquisition (digitization) completes, *i.e.*, after only 33 ms. This disparity is not unique to ActionMedia hardware and is typical of most current hardware that uses discrete cosine transform (DCT) compression schemes and NTSC cameras.

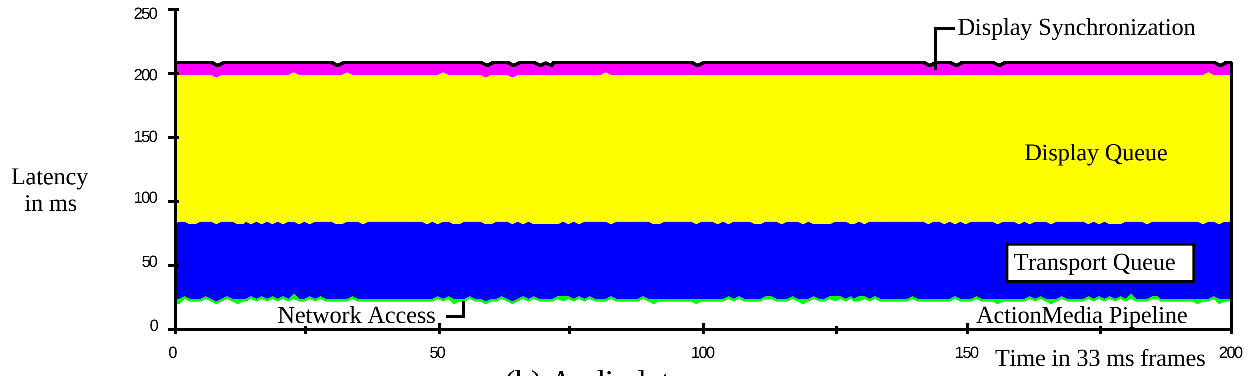
3.2 Representation of system metrics

The most revealing indicator of system performance is the latency of displayed frames plotted over time (see Figure 3.3a). Shown is the latency of a video stream transmitted over a lightly loaded network. In this environment the latency is constant at approximately 210 ms. This figure also shows the relative magnitude of the components of video latency including operating system and application overhead. The components of latency for a single frame are (see also Figures 2.1 and 3.2):

- the time spent in the ActionMedia pipeline at the originating machine,
- the time spent in the transport protocol layer (including time spent in the transport queue),
- the time spent at the network interface waiting to access the network,
- the physical network transmission time,



(a) Video latency.



(b) Audio latency.

Figure 3.3: End-to-end audio and video latency.

- the time spent queued at the receiver waiting to enter the display pipeline, and
- the time spent in the display pipeline.

In Figure 3.3 and throughout, the time spent in the ActionMedia acquisition and display pipelines is shown as a single value since it is essentially constant (approximately 135 ms for 33 ms video frames) and beyond the control of system software. Because each frame may be transmitted as multiple network packets, we also combine the times spent at the network interface and in physical transmission for all packets in a frame into a single measure called *network access time*. This is the total time required to transmit a frame as measured from the time the transport system invokes the network adapter device driver to transmit the first packet containing data for the frame, until the last packet containing data for the frame has been transmitted.

On the receiver, the time spent waiting to enter the display pipeline is further broken down into two components:

- the time spent synchronizing with the display pipeline, and
- the time spent queueing at the entrance to the display pipeline.

We make a distinction between these two components of latency because the former is beyond our control. The clock signal used to initiate a frame display every 33 ms is not synchronized with the clock used to acquire frames on a remote workstation. When a frame arrives at a receiver it may have to wait up to 33 ms before it can be displayed (or more precisely, before it can enter the display pipeline). Therefore, latency may differ between two otherwise identical conferences by as much as 33 ms. The time spent waiting to synchronize with the display pipeline is called the *display pipeline synchronization time* (DPST). In order to compare the effects of transport and display policies across different executions of the system, the DPST must be factored out. The time a frame spends queueing at the entrance to the display pipeline is defined as the time spent queued on the receiver minus the DPST. By definition this time will be a multiple of a frame time. In Figure 3.3a video frames are queued for one frame time at the receiver.

Figure 3.3b shows the relative magnitude of the components of audio latency. Note that audio frames spend a great deal of time in queues waiting to synchronize with video.

Plots of per-frame latency versus time effectively represent both the number of discontinuities in a stream and the synchronization loss between streams. The usual cause of a discontinuity is that frame n+1 is not available when frame n has been completely displayed. In this case, frame *n* is displayed for a second time. When replayed, frame *n* is displayed with one additional frame time of latency (*i.e.*, frame *n* remains in the system for one frame time longer than it otherwise would have). This type of discontinuity manifests itself as an instantaneous increase in latency by one frame time as shown in Figure 3.4. When frame *n*+1 is available but not displayed after frame *n* (*i.e.*, deliberately skipped), the latency in the stream necessarily decreases by one frame time (the frame following the skipped frame is displayed one frame time earlier than it otherwise would have). This type of discontinuity manifests itself as an instantaneous decrease in latency by one frame time as shown in Figure 3.4. The important relation is that discontinuities cause changes in latency and discontinuities occur whenever there is a change in the latency of a displayed stream. Note that latency can only change by multiples of a frame time.

Synchronization loss can be represented in a combined plot of frame latency for corresponding audio and video streams (see Figure 3.4). When audio and video have equal latency they are being played with exact synchronization. When audio has lower latency than video, audio frames are played before their corresponding video frames. When audio has higher latency, audio frames are played after their corresponding video frames. Discontinuities may cause changes in synchronization but discontinuities always occur whenever synchronization changes. In our system, audio and video can only be displayed out of synchronization by multiples of an audio frame time (*e.g.*, by either 16.5 or 33 ms).

4. Transport and Display Mechanisms

We have implemented an unreliable connection-oriented stream transport protocol, called MTP (Multimedia Transport Protocol), on top of UDP/IP. The novel aspect of the protocol is the way data are managed within the transport layer. The design of the

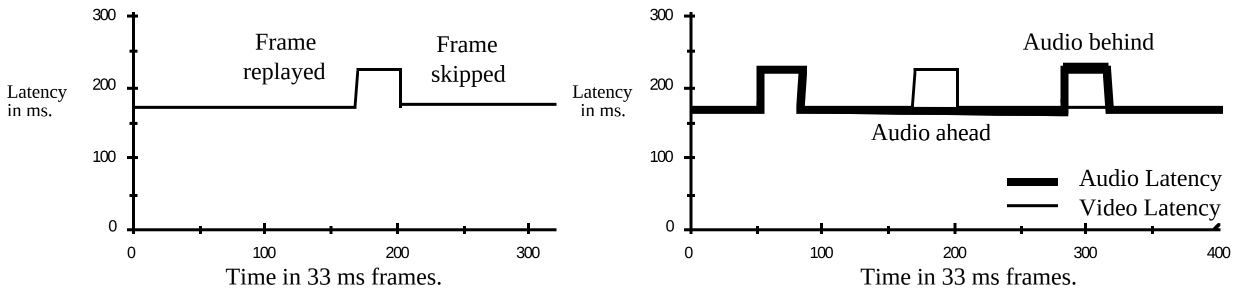


Figure 3.4: A single replay and skipped frame discontinuity in a displayed stream (left). Audio and video latency combined (right).

protocol is based on a real-time systems philosophy. That is, the transport layer uses information on the timing characteristics and semantics of the data to be transmitted so that it can manage the available resources (*e.g.*, CPU cycles, buffers, network transmission slots) to meet the real-time requirements of multimedia applications. This implies that the transport layer is more tightly integrated with higher layers (*e.g.*, the application layer) and lower layers (*e.g.*, the network interface layer) than in traditional protocols. For example, when an MTP connection is established, the application specifies detailed information such as audio/video frame times and maximum tolerable latency.

The essence of this approach is to view the audio/video acquisition processes, the network subsystem, and the display processes as a distributed pipeline (see Figure 2.1). At the sender and receiver, the stages of the ActionMedia pipeline can be tightly synchronized (given sufficient real-time support from the operating system) so that data flows between stages as it is generated (*i.e.*, with 0 or 1 buffers). The network transmission and reception processes cannot be so controlled and hence queues must be maintained in the transport subsystem to interface the network with the audio/video processes. The mechanisms we implement are a set of policies that specify how data is enqueued and dequeued. There are two types of queues to manage: queues at the sender that contain compressed audio/video frames ready for transmission, and queues at the receiver that contain complete audio/video frames that are ready to enter the display pipeline. At both the sender and receiver, separate queues are maintained for audio and video frames.

Given the time-critical nature of audio and video data, dequeueing operations should be performed as late as possible. For example, the ideal time to decide which audio and video should be transmitted next is to wait until access to the network is actually obtained (*e.g.*, when a free token arrives at a station on a token ring). Similarly, the ideal time to decide which frame of video should be displayed is during the vertical blanking interval of the display. Delaying decisions as long as possible maximizes the amount of data that is available to make a decision. The implication of this is that the transport protocol must be integrated with the network device driver on the sender and with the user application at the receiver.

Four mechanisms for real-time transport and display of audio/video data are presented below. For purposes of illustrating the effect of each policy on audio/video fidelity, we use an audio frame time of 33 ms (*i.e.*, audio is acquired at 33 ms intervals) and a video frame time of 66 ms. For video this means that one NTSC frame is acquired during each 66 ms interval. At the receiver each video frame will be displayed for 66 ms. This results in a video frame rate of 15 frames per second and a combined audio/video data rate of approximately 1 megabit/second for a combined audio/video stream. We use a video frame time of 66 ms for the following demonstrations because a longer frame time makes it easier to run controlled experiments.

4.1 Display Queue Management

The goal in display queue management is to ameliorate the effects of network congestion, specifically jitter, to minimize the latency, synchronization differential, and number of discontinuities in the displayed audio/video streams. These goals cannot be achieved simultaneously and hence there are trade-offs between these three measures. In MTP these trade-offs are made through two display queue management techniques: dynamically varying audio/video synchronization and queue length monitoring. We begin with a discussion of the effects of congestion on audio/video latency.

The Effect of Jitter on Audio/Video Latency

Consider the problem of bursty network traffic. Network bursts increase latency and cause discontinuities by introducing jitter. Figure 4.1 shows the effect of a network burst on video latency. Figure 4.1 plots latency versus conference duration (measured in number of frames displayed). Latency is broken down into its constituent components as explained in Section 3.

Video latency is initially 255 ms. After approximately 75 video frames have been generated, load is introduced in the network. As a result, the sender's network access time increases; delaying the arrival of frames at the receiver. The initial increase in network access time is compensated for by a decrease in the DPST at the receiver and thus the latency in the stream remains constant. As frames arrive later and later, the

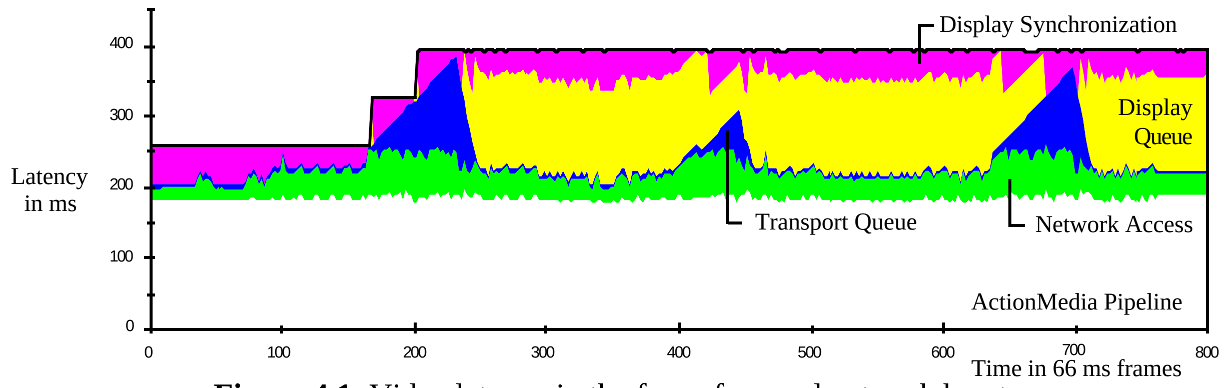


Figure 4.1: Video latency in the face of several network bursts.

DPST eventually becomes 0 indicating that frames are entering the display pipeline immediately upon arrival. At this point the next frame that arrives late (*i.e.*, more than 66 ms after the previous frame) will cause a discontinuity in the video stream (a frame will have to be replayed) and an increase in latency.

As more traffic is introduced into the network, the network access time continues to increase. At frame 170, the access time increases from approximately 30 to 70 ms per frame (recall that multiple network accesses are required to transmit a video frame). This increase results in a queue of unsent frames at the sender indicating that the sender is no longer able to transmit data in real-time. If no action is taken and the load in the network does not subside, frames will be queued at the sender for longer and longer durations. This is illustrated in Figure 4.1 by a linear increase in the time spent in the transport queue starting around frame 170. Eventually a second discontinuity and corresponding increase in latency occurs. By the time the 200th frame is displayed, frames are arriving at the receiver approximately 150 ms after being generated and are displayed with 133 ms more latency than the initial frames.

As the network burst subsides the sender is able to transmit faster than real-time and empties its queue. However, since the receiver can only display frames at a constant rate, a queue forms at the display. Moreover, because the receiver displays frames at a constant rate, the latency in the conference does not change. Thus a temporary increase in the network access time of between 10-15 ms per packet results in a sustained increase in latency of 133 ms. However, as shown in Figure 4.1, the conference is now resilient to future network bursts of equal magnitude. After frame 250, the frames queued at the receiver insulate the display process from additional bursts. In general, the increase in network access time required to induce a discontinuity in a stream will be the sum of the DPST (a random variable) plus the frame time times the display queue length.

Dynamically Varying Audio/Video Synchronization

The network burst in Figure 4.1 increased video latency and created discontinuities. If audio is synchronized with video then the audio stream suffers equally. The effects of bursts on audio can be avoided by exploiting the disparity in the lengths of the audio and video pipelines (and network access times) and allowing the receiver to play audio frames before their corresponding video frames are displayed.

Figure 4.2 shows the effect of a similar network burst on audio latency when audio is synchronized with video. Note that when each discontinuity occurs, audio frames are queued at the display. Audio is queued at the display because each audio frame must

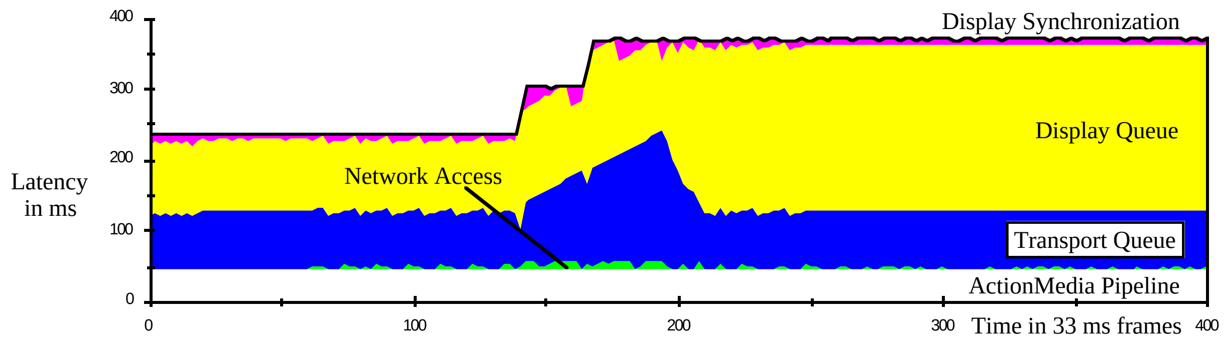


Figure 4.2: Audio latency during a network burst. Audio synchronized with video.

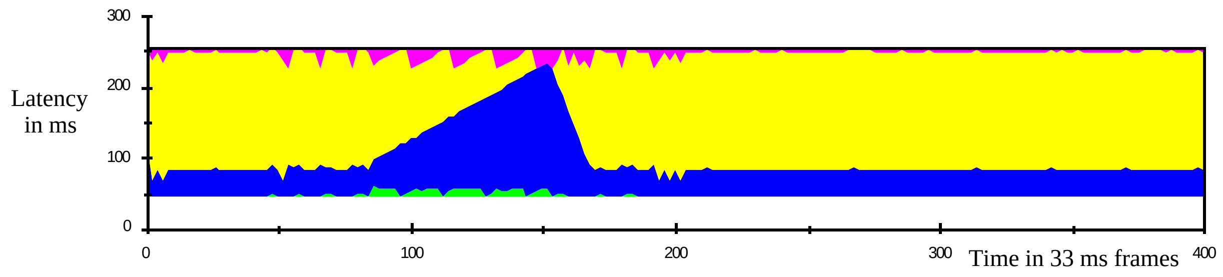


Figure 4.3: Audio latency during a network burst. Audio not synchronized with video.

wait for its corresponding video frame to be decompressed. Thus there are a sufficient number of audio frames in the display queue to play audio without any discontinuities during the burst. This is illustrated in Figure 4.3 (for a comparable size network burst). A combined plot of audio and video latency is shown in Figure 4.4. Note that after a network burst, audio is played with 133 ms less latency than video, *i.e.*, audio is two 66 ms video-frame times ahead of its corresponding video.

By dynamically varying the synchronization between the audio and video streams we are able to minimize discontinuities in the audio stream due to network bursts. This technique has two limiting effects. First, assuming a situation with stable network load and audio/video displayed with minimum possible latency, audio can be played ahead of

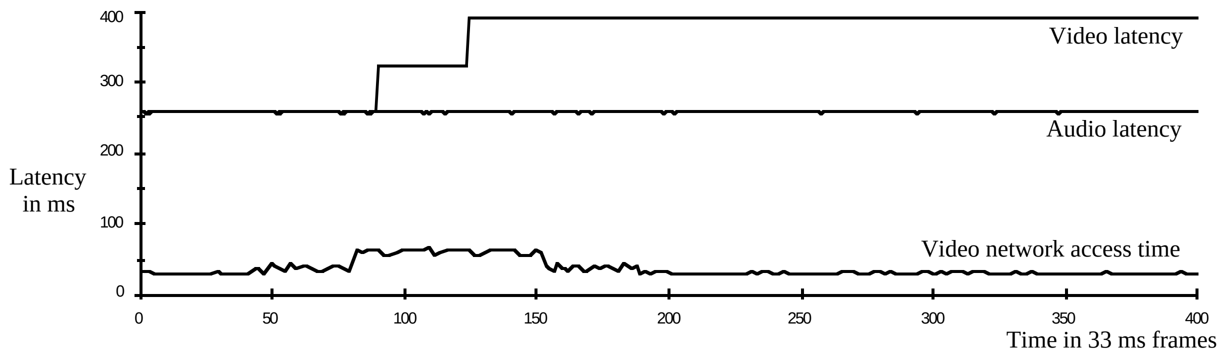


Figure 4.4: Audio and video latency combined during a network burst.

video by no more than the difference in the lengths of the combined acquisition/display pipelines for audio and video plus the difference in the network access times for audio and video frames. In our system, for 33 ms video frames, audio can be played ahead of video by at most 66 ms when the video stream is displayed with minimum latency. Therefore, if audio and video are initially synchronized, the audio stream can tolerate temporary increases in network access times of at least 66 ms per frame (plus the DPST) without creating an audio discontinuity and corresponding increase in latency. If there is additional latency in the video stream due to queueing or high per frame network access (as in Figure 4.1) then audio can be played ahead of video by more than 66 ms.

The second factor limiting the deviation in audio video synchronization is the limit of human perception and tolerance of synchronization between audio and video. Audio should be played ahead of video only to the extent that audio discontinuities are perceived as more annoying the than loss of audio/video synchronization. For full-motion video, we conjecture that users will tolerate audio being played by 100 ms ahead of video [18]. Informal user studies indicate that this is the level at which the loss of synchronization is first noticeable in our system (given a particular video compression algorithm and its resulting image quality).

Queue Length Monitoring

The second display queue management technique reduces latency in a stream following periods of congestion. After the initial network burst shown in Figure 4.1, video is displayed with an additional 133 ms worth of latency. After frame 250, the additional latency results in a queue of frames at the receiver. Because frames enter the queue at the same rate at which they are played, latency can be reduced only by discarding a frame and playing a newer frame in its place. For example, in Figure 4.5, after 240 video frames have been displayed, a frame is discarded, shortening the display queue by one frame and reducing the latency by 66 ms. However, reducing latency reduces the ability of a display to tolerate future bursts in network load. The receiver can therefore choose to either minimize latency or avoid discontinuities caused by future network bursts. To manage this trade-off we use a technique called *queue length monitoring* to estimate the current load in the network and display audio and video with latency that is appropriate for the current network conditions.

We associate an array of counters and threshold values with the display queue. When a remove operation is performed, if the display queue has length $m > 0$, then counters 1 through m are incremented; all other counters are reset. For each counter that exceeds its

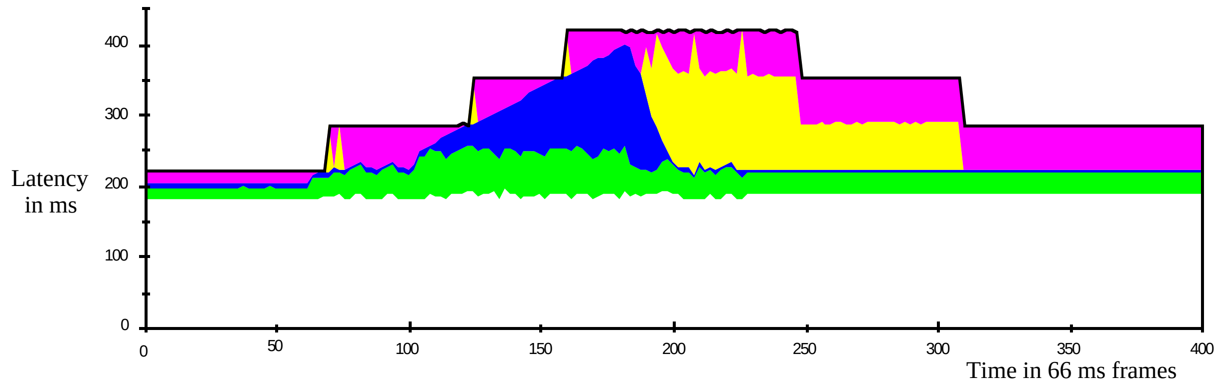


Figure 4.5: Video latency with display queue length monitoring.

threshold value, the counter is reset and one frame is discarded from the queue (in FIFO order). The oldest remaining frame is returned as the result of the remove operation.

For counter *n*, the threshold value specifies a duration (measured in frame times) during which the display queue must contain at least *n*+1 frames. If this is the case then we conclude that the jitter in the arriving stream is low enough to maintain a conference with lower latency. Currently, threshold values for large queue lengths specify small durations (*e.g.*, half a second) and increase geometrically for smaller queue lengths. This is done to reduce latency quickly after large bursts of network traffic (which we assume occur infrequently) but to approach minimal latency slowly so as to avoid discontinuities caused by smaller, more frequent, variations in traffic.

Figure 4.5 shows an example of display queue length monitoring for the video queue. At approximately frame 180, the network burst subsides. After the burst, significant jitter persists as seen by fluctuations in the times frames spend in the display queue. However, during the display of frames 180-240, the display queue always contains at least two frames. Therefore, at frame 240 it is decided that the display queue can be shortened by one frame without adverse effect. After frame 240 the network is relatively idle and there is virtually no jitter. At frame 310 the display queue is again shortened by one frame and video is displayed with minimal latency.

4.2 Transport Queue Management

Display queue management ameliorates the effects of network congestion at the receiver. Transport queue management ameliorates the effects of network congestion at the sender. At the sender, network congestion results in varying network access time which results in variation in the bandwidth available to the sender for transmitting audio/video frames. In addition, congestion can lead to packet loss in the network. The goals in transport queue management are to (1) adapt the outgoing frame rates to the bandwidth currently available to the sender, (2) deliver video frames with minimal latency, and (3) reliably deliver audio frames. Again, there are trade-offs between these goals.

Two transport queue issues are discussed. The first is a choice of queue length for the audio/video transport queues. The second is the algorithm used for combining audio and video frames into network packets. This includes a forward error correction scheme that transmits audio frames multiple times to ameliorate the effect of packet loss in the network.

Queue Length

Audio/video frames are delivered to the transport system by the application using a non-blocking (asynchronous) send operation. To maintain maximal frame rate, on average, one frame must be transmitted during each frame time. When the network access time becomes larger than the frame time, queues form in the transport system. If multiple frames can be transmitted in a single network packet, or if multiple frames may be transmitted during a frame time in the near future, then maintaining a queue is desirable since it increases the likelihood that all frames will eventually be transmitted. However, enqueuing frames that cannot be transmitted immediately increases the latency of the conference. The choice of a length for the queues in the transport system is thus a trade-off between throughput (frame rate) and latency.

Since this trade-off should be made based on application requirements, the choice of queue length is a parameter of MTP. Theoretically, a transport queue of length n implies that the sender can tolerate a one-time maximum network access time of at most ($n \times 1$ frame time) time units and still maintain the maximum frame rate. However, allowing a queue of length n also implies that in times of severe network congestion one is willing to tolerate as much as ($n \times 1$ frame time) additional time units of latency in the displayed stream. Moreover, once the network access time falls below a frame time, it will take at least time proportional to the queue length to return the transport queue component of latency to a minimum.

Separate queues are maintained for audio and video frames at the sender. For video, we choose to minimize latency and thus use a transport queue of length one. This is primarily because latency is more important than frame rate for live, interactive video. (In particular, since audio and video are synchronized, maximizing video latency adversely effects audio latency.) A second, more pragmatic reason, is that with our current hardware, it is not possible to transmit substantially more than a single video frame during a frame time. Therefore, using a transport queue of length greater than one has no effect on video throughput.

Given our goal of reliable transmission of audio, we choose to maximize audio throughput and use a queue sufficient to hold 1 second of audio data. For the networks we consider, multiple audio frames can be transmitted in a single packet and thus there is a significant benefit to maintaining a large audio queue.

When queues in the transport system are full, frames are discarded in FIFO order. Should this occur, it indicates that for the particular trade-off between latency and throughput made by the application, there is insufficient bandwidth for maintaining the maximum frame rate. In our network environment, this situation can be overcome only by the application. The application can either allow additional latency in the effected stream or it can reduce the bandwidth requirements of the stream by changing the coding and/or compression scheme. When queues overflow, this information is fed back to the application so that it may take steps to remedy the situation.

Packetizing Audio and Video

Because of congestion in the network, it is possible that audio/video frames may not arrive at the receiver. Frames may not arrive because they were never transmitted (*i.e.*, discarded at the sender) or because packets containing portions of the frame were lost in the network. The former case indicates that there is insufficient bandwidth for maintaining the maximum frame rate. The latter case can be overcome by re-transmitting lost frames. Traditional methods for re-transmission use a time-out mechanism. This is not practical in our environment as the latency incurred for the time-out and re-transmission would be unacceptable. Instead we adopt a forward error correction strategy and transmit frames multiple times. An application defined portion of every packet is reserved for redundant transmission of audio frames. For video we do not attempt to ameliorate the effects of packet loss.

The basic packetizing algorithm used in the transport layer is to place audio and video frames into a network packet according to their deadlines. Each frame has a deadline equal to its generation time plus one frame time. Packets are then scheduled at the network interface using a deadline-based scheduling algorithm [12].

Once an audio frame has been transmitted it is placed on a re-transmission queue. Frames from this queue are used to fill the forward error correction portion of every MTP packet. Two parameters, a minimum and maximum transmission separation, drive this process. The minimum separation parameter establishes a lower bound on the time between successive transmissions of the same frame. A minimum separation of *t* frames ensures that each re-transmission of the same audio frame is separated by the transmission at least *t* (original) audio frames. The maximum separation parameter is an upper bound on the time between successive transmissions of the same frame. A maximum separation of *t* frames ensures that if re-transmission of an audio frame occurs, it occurs no later than *t* frame times after the frame was originally generated. The maximum separation is chosen so that in the event that the packet containing the original audio frame is lost, the re-transmitted frame will arrive in time to be played in the place of the original. These minimum and maximum separation times result in most audio frames being re-transmitted once. This increases the total bandwidth requirements for a conference by approximately 10%. Experiments indicate that this simple re-transmission of audio frames is extremely effective.

5. Performance

In this section we demonstrate empirically the effectiveness of the transport and display mechanisms for conferences spanning a collection of token rings and ethernets (see Figure 3.1). Each experiment below compares the performance of MTP to a straightforward use of UDP for three measures of conference performance: throughput (frame rate), latency, and synchronization of audio and video. All experiments use an audio frame time of 16.5 ms and a video frame time of 33 ms. This results in a maximum data rate for a combined audio/video stream of just under 2 megabits per second. To stress the network, we artificially generate load on all networks by simulating bulk data transfers.

In the MTP conferences, the transport queue for the video connection holds only one frame. The audio queue can hold 60 frames (1 second of audio data). The portion of each packet reserved for forward error correction is large enough to hold 3 audio frames. The minimum and maximum separation parameters for audio forward error correction are 1 and 8 audio frame times respectively. This means that after an audio frame is transmitted, it is buffered for at least 16.5 ms before it is eligible to be sent a second time and discarded 133 ms after it was generated. These settings result in most audio frames being sent a total of two times. At the receiver, audio is allowed to get ahead of video by at most 100 ms. When the network is lightly loaded, MTP maintains 30 video frame per second conferences with average video latency of 200 ms (plus or minus the 33 ms random variation in DPST) while audio throughput is 60 frames per second with average latency of 100 ms.

In the UDP conferences, the conferencing application always transmits audio and video frames in separate packets (since audio and video are acquired by separate processes). Multiple audio frames are placed in a single UDP message. At the receiver, audio and video are always played in exact synchronization. On lightly loaded networks our use of UDP suffices to sustain conferences with a frame rate of 30 video frames per second, 60 audio frames per second, and average audio/video latency of 200 ms (plus or minus the 33 ms random variation in DPST).

For both the UDP and MTP conferences, Figure 5.1 summarizes the number of frames transmitted, displayed, lost in the network, and discarded by either the sender or the receiver. Figures 5.2 and 5.3 show audio and video frame rate and latency for the UDP and MTP protocols respectively. Qualitatively, the MTP conference was acceptable. The average audio latency, approximately 240 ms, is under the 250 ms target. The playout of audio was clear and continuous with numerous 20-30 second intervals during which there were no discontinuities. When discontinuities occurred they were sufficiently separated in time so as not to be annoying. Recorded music was used for the audio component of the conference.[1] For example, many discontinuities were purposely caused by the queue length monitoring mechanism at the receiver in an attempt to reduce audio latency. These discontinuities occurred when the display queue (and hence the network) was relatively stable and thus were isolated events. The average MTP video latency was approximately 330 ms and varied greatly. The displayed video was reasonably smooth although there were several instances where video discontinuities clustered together and resulted in image “freezes” and “jerks.”

Qualitatively, the UDP conference was not acceptable. Because of persistent discontinuities, the audio was nearly unintelligible. Video throughput was 15% less than in the MTP case, however the perceived quality was comparable to the MTP video. Average video latency was approximately 340 ms and only slightly more jerky than in the MTP case.

[1] It is easy to hear discontinuities in music and hence for the purposes of assessing the performance of the transport protocol, music provides a more challenging audio source than speech.

	Sender			Network	Receiver			Averages	
	Generated	Dropped	Sent	Lost	Received	Dropped	Played	Frame Rate	Latency
UDP	17,999	2,555	15,444	142	15,302	0	15,302	24.9	343
MTP	17,999	17	17,982	156	17,826	0	17,826	29.7	333

(a) Video frames.

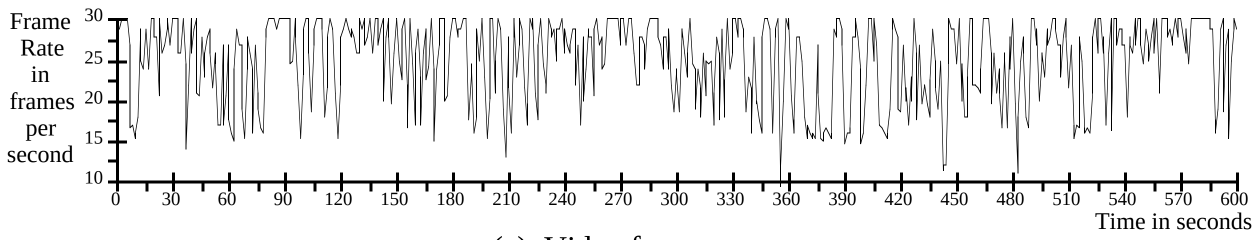
	Sender			Network	Receiver				Averages	
	Generated	Dropped	Sent	Lost	Received	Recovered	Dropped	Played	Frame Rate	Latency
UDP	36,000	0	36,000	563	35,437	0	3,913	31,424	52.3	341
MTP	36,000	0	36,000	214	35,786	173	125	35,834	59.6	238

(b) Audio frames.

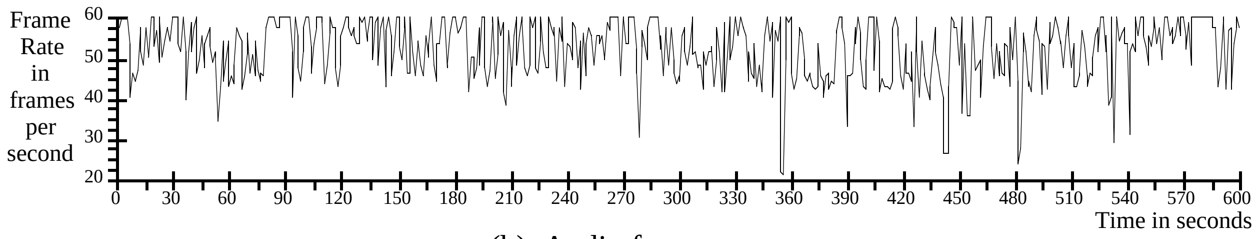
Figure 5.1: Throughput and latency statistics.

Audio in the UDP conference suffered because the application attempted to play audio and video in exact synchronization and because the audio and video were transmitted as separate streams. The effect of the latter is that in the UDP conference, the sender requires many more network accesses than in the MTP conference. This means that in times of congestion UDP will require more tokens on the token ring than MTP and will create more load on bridges and routers in the network. Because of this, several thousand video frames were never transmitted because UDP queues overflowed before the video could be transmitted. This negatively impacts the playout of audio since playing audio and video in exact synchronization is problematic when video frames either do not arrive or arrive late. When video frame *n* does not arrive on time the display application does not play the corresponding audio and waits for the late frame. If frame *n*+1 arrives before *n*, the audio for the frame *n* is discarded and frame *n*+1 and its corresponding audio are displayed. Poor video throughput and the exact audio/video synchronization requirement result in the dropping of several thousand audio frames at the receiver. Moreover, because the UDP conference physically transmitted more packets, it also experienced a higher rate of packet loss. This is in spite of the fact that it transmitted fewer bytes of data then the MTP conference.

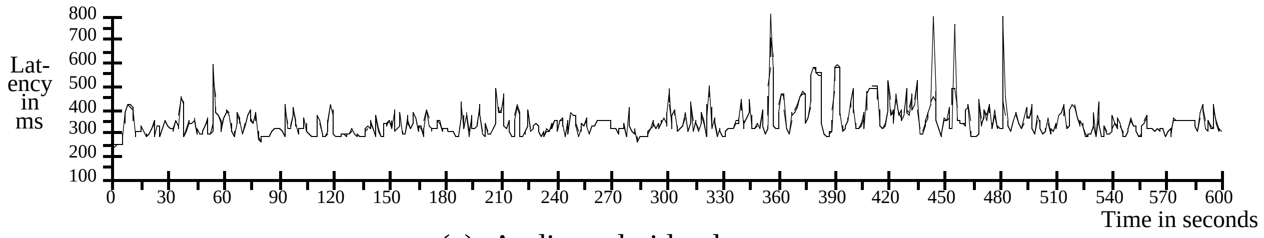
Our use of the transport and display mechanisms worked well. The MTP conference had better audio throughput because the transport mechanisms dynamically combine audio and video into network packets thus reducing the number of network accesses required to transmit audio and video. Video playout was slightly less jerky than in the UDP conference because the MTP video did not experience large instantaneous increases in latency as did the UDP video. This was due in part to the fact that in times of severe



(a) Video frame rate.

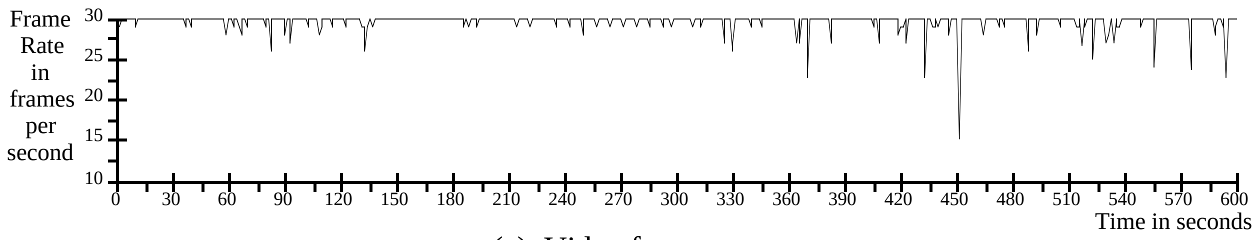


(b) Audio frame rate.

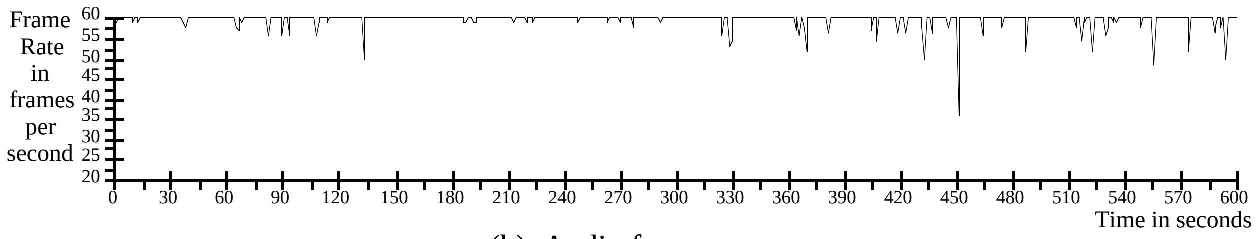


(c) Audio and video latency.

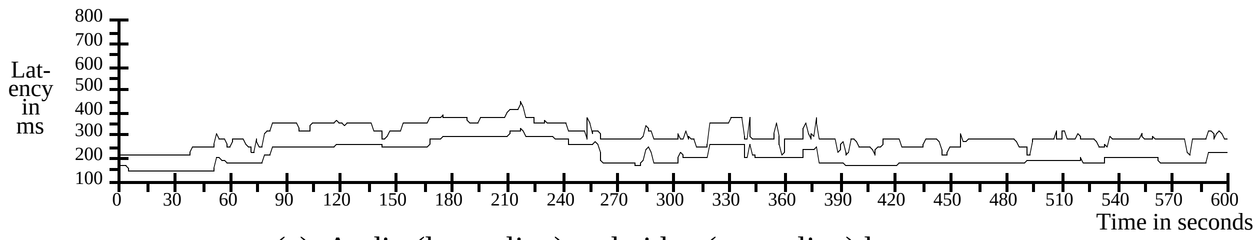
Figure 5.2: UDP performance.



(a) Video frame rate.



(b) Audio frame rate.



(c) Audio (lower line) and video (upper line) latency.

Figure 5.3: MTP performance.

congestion on the token ring, MTP did not queue video frames at the sender. Because UDP queued video frames the effects of increases in network access time were magnified by the existence of the longer queue.

In periods of high jitter and widely varying video latency, varying audio/video synchronization resulted in stable, discontinuity-free audio. The forward error correction scheme recovered 80% of the audio frames lost in the network. At the receiver, queue length monitoring resulted in the dropping of 125 audio frames to reduce the latency of the audio stream. Given the high level of congestion in the network, there was never a sufficient amount of video queued at the receiver for the queue length monitoring to have an effect on the video stream.

The display mechanisms are primarily concerned with ameliorating the effects of jitter and hence their ability to work well is a function of the jitter in the arriving streams. In particular, the performance of the display mechanisms is not a function of the number of networks spanned by the conference — only the jitter induced by the aggregation of the networks. In contrast, the transport mechanisms are primarily concerned with packaging audio and video frames to best utilize the bandwidth available to the conference as seen by the sender. In the current implementation of MTP, the sender is only able to estimate the available bandwidth on the network to which it is directly attached. Although they worked well in the above experiment, the transport mechanisms do not scale with the number of networks spanned by a conference.

When the preponderance of the network traffic is on the sender’s local network, the transport and display mechanisms are extremely effective. Figures 5.4 through 5.6 compare the performance of UDP and MTP respectively, for a conference on a single token ring. In this experiment traffic was artificially placed on the network to induce periods of network utilization in excess of 99%. This resulted in both high jitter and periods of time in which it was not possible to sustain a video throughput of 30 frames per second.

The UDP conference was better than in the previous experiment, primarily because there were several periods in which the network was sufficiently idle to achieve full audio throughput with little jitter. On the whole, however, the audio was unintelligible. Video was of comparable quality to that in the first experiment. In contrast, the MTP conference was nearly flawless. The few audio discontinuities that occurred were sufficiently separated in time so as not to be noticeable. The video appeared “normal.”

	Sender			Network	Receiver			Averages	
	Generated	Dropped	Sent	Lost	Received	Dropped	Played	Frame Rate	Latency
UDP	17,999	3,878	14,121	0	14,121	0	14,121	23.4	283
MTP	17,999	530	17,469	1	17,468	6	17,462	29.1	223

(a) Video frames.

	Sender			Network	Receiver				Averages	
	Generated	Dropped	Sent	Lost	Received	Recovered	Dropped	Played	Frame Rate	Latency
UDP	36,000	635	35,365	0	35,365	0	4,800	30,565	50.69	281
MTP	36,000	0	36,000	2	35,998	1	5	35,994	59.98	143

(b) Audio frames.

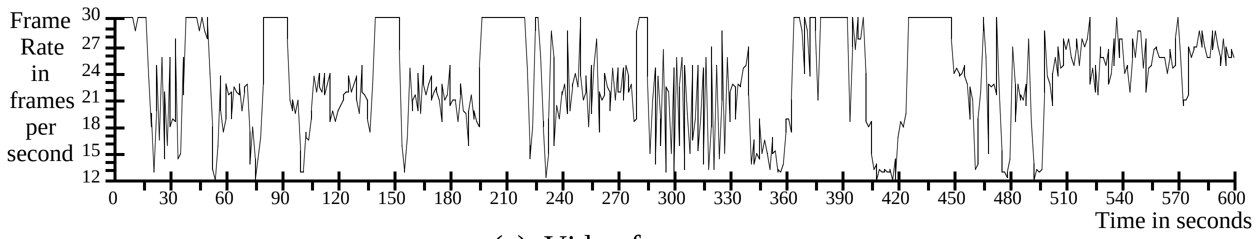
Figure 5.4: Throughput and latency statistics, high token ring traffic.

We conclude that the transport mechanisms can be made to scale with the number of networks spanned by a conference by providing the sender with information on the load in network along the path from sender to receiver. To do this we are investigating a feedback mechanism for propagating bandwidth information (measured in terms of sustainable frame rate) for use in conjunction with the transport mechanisms.

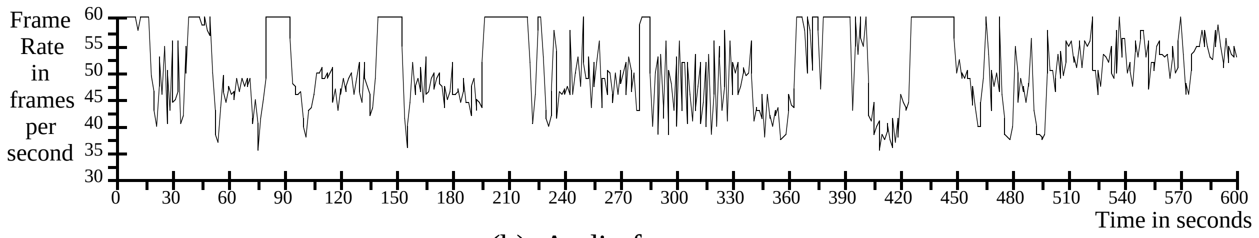
6. Summary and Future Work

The problem space of computer systems support for digital audio and video is large and diverse. We are working in a region of this space that is characterized by video that is digitized and compressed on NTSC frame boundaries and by interconnected, asynchronous, local area networks. Our goal is to support computer-based videoconferencing across small networks using commodity audio/video technology and without special purpose network services. This work is important because even in the presence of special network services such as resource reservation and admission control, today's conventional LANs are likely to persist for some time. We believe this is particularly true for those LANs connecting individuals' workstations to larger internetworks. The fundamental problems addressed herein, namely the problem of ameliorating jitter, load variation, and packet loss, will remain in these environments. Any solution must address these problems to provide low latency, synchronized audio and video communications.

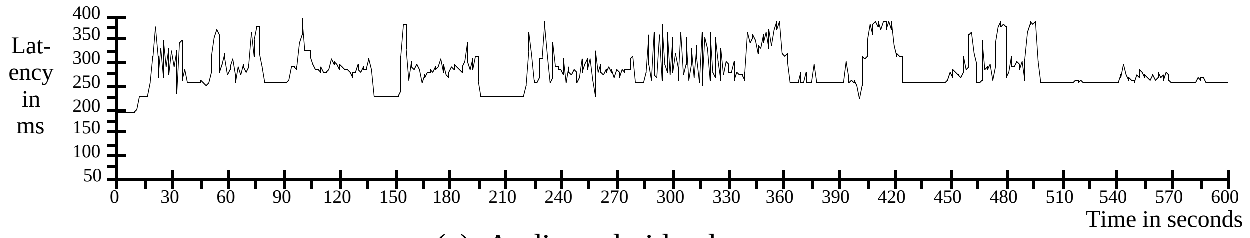
We have developed a multimedia transport protocol, MTP, that attempts to support high performance conferences in the presence of congestion. Our primary measures of conference performance are audio/video latency, frame rate, and synchronization. MTP



(a) Video frame rate.

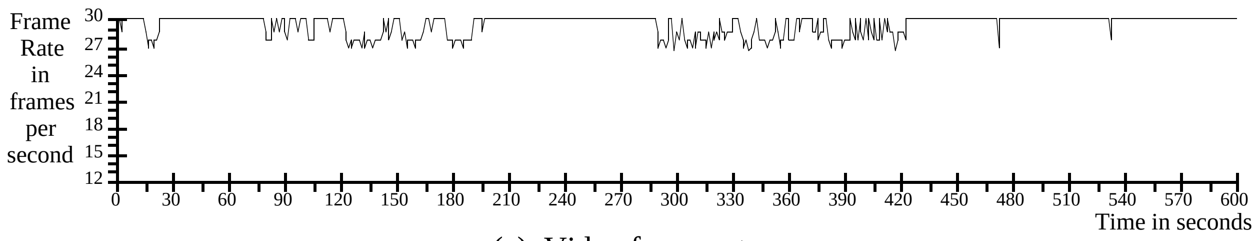


(b) Audio frame rate.

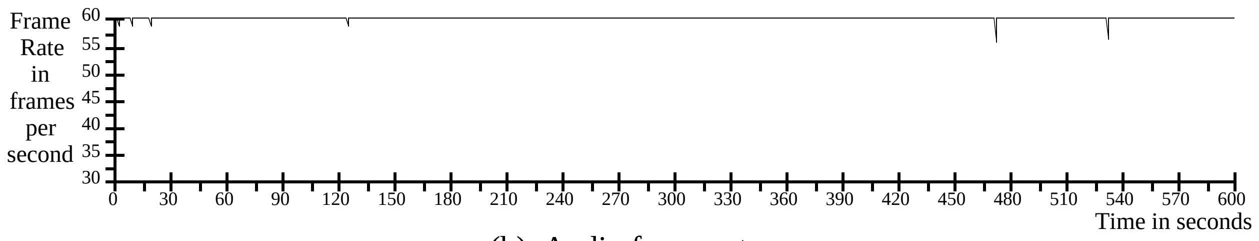


(c) Audio and video latency.

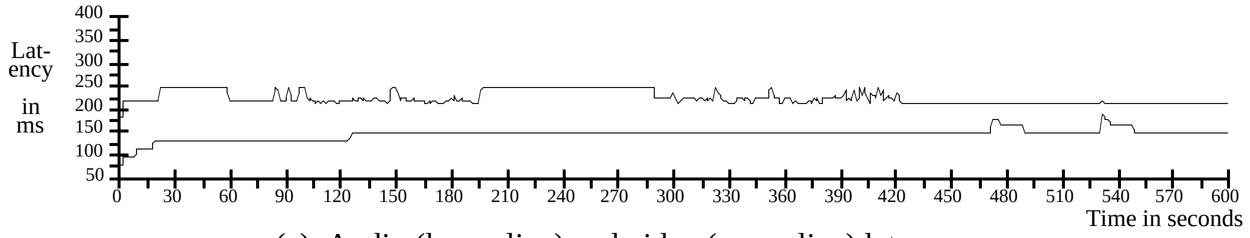
Figure 5.5: UDP performance, high token ring traffic.



(a) Video frame rate.



(b) Audio frame rate.



(c) Audio (lower line) and video (upper line) latency.

Figure 5.6: MTP performance, high token ring traffic.

provides “best effort” delivery of audio and video data through four transport and display mechanisms and a real-time implementation of these mechanisms that integrates operating system services (*e.g.*, scheduling and resource allocation, and device management) with network communication services (*e.g.*, transport protocols) and applications. The four mechanisms are: a facility for varying the synchronization between audio and video streams to achieve continuous audio playout in the face of jitter, a network congestion monitoring mechanism that is used to control audio/video latency, a queueing mechanism at the sender that is used to maximize frame throughput without unnecessarily increasing latency, and a forward error correction mechanism to ameliorate the effects of packet loss in the network. In the presence of congestion, these mechanisms are capable of dynamically adapting the conference frame rate to the bandwidth available in the network, minimizing the latency in the audio/video streams while avoiding discontinuities, and providing quasi-reliable delivery of audio frames.

Much work remains. While the effectiveness of the transport and display mechanisms have been demonstrated in isolation and in concert, our understanding of how best to employ these mechanisms is incomplete. Each mechanism is characterized by several parameters. At the sender, MTP parameters include audio and video queue lengths, percentage of each packet that is reserved for forward error correction. At the receiver, parameters include synchronization limits, and threshold values for audio and video display queues. Appropriate choices for these parameters as well as interactions between parameter settings is the subject of future study.

7 References

[1] Abdel-Wahab, H., Feit, M. A., 1991. *XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration*, Proc., IEEE Conf. on Communications Software: Communications for Distributed Applications & Systems, Chapel Hill, NC, April 1991, pp. 159-167.

[2] Anderson, D.P., Herrtwich, R.G., Schaefer, C., 1990. *SRP: A Resource Reservation Protocol for Guaranteed Performance Communication in the Internet*, University of California Berkeley, Dept. of Electrical Eng. and Computer Science Technical Report, TR-90-006, February 1990.

[3] Anderson, D.P., Tzou, S.-Y., Wahbe, R., Govindan, R., Andrews, M., 1990. *Support for Continuous Media in the DASH System*, Proc. Tenth Intl. Conf. on Distributed Computing Systems, Paris, France, May 1990, pp. 54-61.

[4] Ferrari, D., 1990. *Client Requirements for Real-Time Communication Services,* IEEE Communications, (November), pp. 65-72.

[5] Ferrari, D., 1991. *Design and Application of a Jitter Control Scheme for Packet-Switch Internetworks*, Proc. 2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Germany, pp. 81-84.

[6] Govindan, R., Anderson, D.P., 1991. *Scheduling and IPC Mechanisms for Continuous Media*, Proc. ACM Symp. on Operating Systems Principles, ACM Operating Systems Review, Vol. 25, No. 5, (October), pp. 68-80.

[7] Harney, K., Keith, M., Lavelle, G., Ryan, L.D., Stark, D.J., 1991. *The i750 Video Processor: A Total Multimedia Solution*, Comm. of the ACM, Vol. 34, No. 4 (April), pp. 64-79.

[8] Hopper, A., 1990. *Pandora — An Experimental System For Multimedia Applications*, ACM Operating Systems Review, Vol. 24, No. 2, (April), pp. 19-34.

[9] Ishii, H, Miyake, N., 1991. *Toward an Open Shared Workspace: Computer and Video Fusion Approach of Team Workstation*, Comm. of the ACM, Vol. 34, No. 12 (December), pp. 37-50.

[10] Jeffay, K., Stone, D., Poirier, D., 1992. *YARTOS: Kernel support for efficient, predictable real-time systems*, in “Real-Time Programming,” W. Halang and K. Ramamritham, eds., Pergamon Press, Oxford, UK.

[11] Jeffay, K., Stone, D.L., and Smith, F.D., 1992. *Kernel Support for Live Digital Audio and Video*. Computer Communications, Vol. 16, No. 6 (July), pp. 388-395.

[12] Jeffay, K., 1992. *Scheduling Sporadic Tasks with Shared Resources in Hard-Real-Time Systems*, Proc. 13th IEEE Real-Time Systems Symp., Phoenix, AZ, December 1992, pp. 89-99.

[13] Jeffay, K., Lin, J.K., Menges, J., Smith, F.D., Smith, J.B., 1992. *Architecture of the Artifact-Based Collaboration System Matrix*, CSCW ‘92, Proc. of the Conf. on Computer-Supported Cooperative Work, Toronto, Canada, ACM Press, November 1992, pp. 195-202.

[14] Katseff, H.P., Gaglianello, R.D., London, T.B., Robinson, B.S., Swicker. D.B., 1990. *An Overview of the Liaison Network Multimedia Workstation*, Proc. First Intl. Workshop on Network and Operating System Support for Digital Audio and Video, Berkeley, CA.

[15] Le Gall, D., 1991. *MPEG: A Video Compression Standard for Multimedia Applications*, Comm. of the ACM, Vol. 34, No. 4, (April), pp. 46-58.

[16] Liou M., 1991. *Overview of the px64 kbit/s Video Coding Standard*, Comm. of the ACM, Vol. 34, No. 4, (April), pp. 59-63.

[17] Rangan, P.V., Vin, H.M., 1991. *Designing File Systems for Digital Video and Audio*, Proc. ACM Symp. on Operating Systems Principles, ACM Operating Systems Review, Vol. 25, No. 5, (October), pp. 81-94.

[18] Steinmetz, R., Meyer, T., 1992. *Multimedia Synchronization Techniques: Experiences Based on Different System Structures*, IEEE Multimedia Workshop, Monterey, CA, April, 1992.

[19] Wallace, G.K., 1991. *The JPEG Still Picture Compression Standard*, Comm. of the ACM, Vol. 34, No. 4, (April), pp. 30-44.

[20] Watabe, K., Sakata S., Maeno K., Fukuoka H., Ohmori T., 1990. *Distributed Multiparty Desktop Conferencing System: MERMAID*, Proceedings, CSCW 90 Conference on Computer-Supported Cooperative Work, October 1990.