A Transmission Control Framework for Continuous Media

by
Terry Michael Talley

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
1997

Approved by:

_____
Advisor: Kevin Jeffay

_____
Reader: F. Donelson Smith

_____
Reader: Bert J. Dempsey

ABSTRACT

A Transmission Control Framework for Continuous Media

(Under the direction of Kevin Jeffay)

Desktop video conferencing allows people to simulate face-to-face conversations by integrating real-time two-way audio and video with the computer system. Unfortunately, the quality of video conferences carried over current networks such as the Internet is often inadequate for effective communication. Network congestion can cause video conferences to experience high latencies and poor fidelity. We claim that in many cases we can sustain low-latency, high-fidelity conferences over current networks even when the networks are highly congested if we carefully manage the transmission of the audio and video streams at the endpoints of the conference.

Network congestion is caused by two distinct types of network constraints: *capacity constraints* and *access constraints*. Capacity constraints limit the bit rate that can be supported by the network. Access constraints limit the message rate that can be supported by the network. We claim that conferences can heuristically identify the type of network constraint causing congestion and ameliorate the effects of the congestion by carefully selecting the bit and message rates associated with each of the conference media streams. We explain and empirically demonstrate why addressing capacity and access constraints requires two complementary transmission adaptations: *scaling* and *packaging*. Scaling increases or decreases the bit rate associated with a media stream by controlling the generation and compression of the media stream. Packaging increases or decreases the conference message rate by controlling the type and number of media data units, or frames, placed into each message. We describe a transmission control framework that shows how scaling and packaging can be used to preserve conference quality when the network has capacity constraints, access constraints, or a combination of capacity and access constraints. We develop a transmission control algorithm based on this framework and demonstrate that the algorithm can deliver low-latency, high-fidelity conferences even on heavily congested networks. Finally, we show that adaptation of both the bit and message rates produces conferences with lower latency and higher fidelity than those produced by non-adaptive transmission algorithms or those produced by commonly used algorithms that only scale the video bit rate.

To Brenda and Drew

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

Chapter I
Introduction

Low cost computers and the Internet have revolutionized the way people use computers. Before the widespread availability of low cost computers, only large government and business organizations could afford the cost of computer systems. The arrival of personal computers in the late 1970s changed this situation. For the first time, computers were available to large numbers of people. The demands of this large group of computer users has led to an explosion in the number and diversity of computer applications. One of the most important of these applications is the use of the computer as a communications tool.

In the early 1980s, most computers were not connected to a network and people usually communicated or collaborated outside the computer systems. In the 1990s, the rapid expansion of networks, particularly the Internet, changed the typical environment. Now most computers are, or can be, connected to a network and can potentially exchange information with other computers. Communication is a fundamental human activity and the availability of an inexpensive and widely available communications path between computers has resulted in the computer becoming an integral part of the communications process. For many people, the computer is now the tool of choice for correspondence (via electronic mail), news and information gathering (via the World Wide Web), and group discussion (via news groups). In addition, computer-based communication has become significantly richer in recent years due to the increasing support for digital audio and video in personal computers. CD-ROM drives and sound cards are now standard equipment and digital video cards providing hardware support for video compression schemes such as MPEG [69] are becoming increasingly common. Applications such as computer games, on-line encyclopedias, and web browsers now routinely use multiple media forms -- audio, video, text, graphics, *etc*. -- in concert to increase the appeal and utility of the application.

Unfortunately, most forms of computer-based communication are not conversational. Two individuals communicating via computer usually do not interact as if they were in a face-to-face conversation. For example, when communicating via electronic mail there is typically a relatively long period of time between sending a message to someone and receiving their response. Ideally, computer-based communications would support the same interactive conversational mode used when people talk face-to-face. To do this, computer communications must support both sound and sight and there must be low transmission delay among the participants in the conversation. In face-to-face conversations, sound is the dominant information carrier and we expect a relatively short delay between transmitting the message and receiving the response. Sound is augmented by sight. People often convey confusion, annoyance, humor, attention, *etc*. by their facial expressions and body language. We use visual cues to illustrate points, provide feedback, control conversational turn-taking, convey attitudes, and deal with conflict [34, 51]. We gesture, nod or shake our heads, seek or avoid eye contact, smile, frown, or look bewildered. Speakers are adept at adjusting their speech content to the level of understanding of the listener, as indicated by visual clues, and speakers expect different levels of feedback depending upon the complexity of the topic [51]. When only audio is available (*e.g.*, as in a telephone call) people must compensate for the lack of vision by providing more verbal feedback for attentiveness, comprehension, *etc.*, with the result that "video interactions were markedly richer, subtler, and easier than the telephone interactions" [51]. People perceive audio-only discussions to be less spontaneous, more formal, more socially distant, and less interactive than face-to-face discussions [98]. For these reasons, we believe that video conferencing is a useful and desirable enabling technology, but we recognize that not everyone shares this view [80] and that the telephone is still the most ubiquitous and satisfying communications tool currently available.

The goal of video conferencing is to extend computer-based communication to approximate face-to-face conversations by allowing people to hold low delay, two-way conversations with both audio and video. We are interested in video conferencing where the conferencing facilities are tightly integrated with the participants' computers. This integration allows people to collaborate not only using audio and video, but also using collaborative computer applications such as shared word processing or shared drawing applications [58]. This type of conferencing is sometimes called *desktop video conferencing* to distinguish it from dedicated, stand-alone conferencing systems. We use the terms *video conferencing* and *desktop video*

conferencing interchangeably throughout this dissertation. Desktop video conferencing is still in its infancy and many technical problems inhibit the widespread use of conferencing systems. This dissertation describes some of these problems and particularly focuses on how to avoid or ameliorate the problems associated with transmitting audio and video across computer networks. Before introducing the specific problems and our solutions to those problems, we first describe the elements of a typical video conferencing system.

1.1. Elements of a Video Conferencing System

Figure 1-1 shows the major components of a desktop video conferencing system. A video conference has two or more participants each using a computer to (1) capture and transmit audio and video to other conference participants, and (2) receive and play the audio and video sent from other participants. Each computer is equipped with (1) input and output devices to capture and display audio and video; (2) device controllers that connect the input and output devices to the computer; (3) audio and video encoders and decoders; (4) a network service that connects the computer to a network and allows applications to send and receive data across the network; and (5) a conferencing application that controls the acquisition, transmission, synchronization, and playing of audio and video. Each computer has an operating system that manages the sharing and competition for resources such as CPU or memory by applications running in the computer. We discuss each of these components in more detail below.

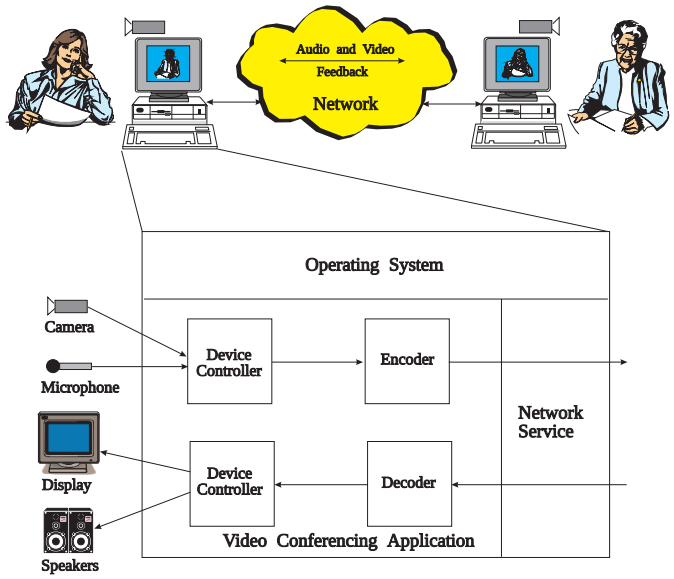


Figure 1-1: Components of a Desktop Video Conferencing System

1.1.1. Input and Output Devices

Most people are familiar with common video conferencing input devices such as video cameras, microphones, and CD players and output devices such as video displays and audio speakers. The role of the input devices is to capture sight and sound by converting light or sound waves into some analog or digital representation. The role of the output devices is to *play* the captured input signal by converting some analog or digital representation of the input signal into light and sound waves. We use the terms *display* and *play* interchangeably to describe output of either audio or video.

Fidelity is the degree to which the output of a system accurately reproduces the characteristics of the original input signal. The quality of the input and output devices affects the fidelity of played images and sounds. For example, in a stereo system the fidelity of played music is better when the input device is a compact disc player than when the input device is a phonograph and most people can perceive the fidelity difference when listening to the music. Similarly, output devices such as the stereo speakers affect the fidelity and perceived quality of music. As implied by the stereo example, there is a strong relationship between fidelity and the perceived quality of played media. We discuss this relationship in more detail later, but better fidelity generally implies better perceived quality (up to the limits of human perception).

In a video conferencing system, fidelity may be lost not only due to limitations of the input and output devices, but also due to the effects of manipulation and transmission of the input data. Since this dissertation focuses on software techniques used by the video conference application, we largely ignore poor fidelity resulting from the input and output devices. On the other hand, the conferencing software has some control over the fidelity lost due to manipulation and transmission of the audio and video data and controlling this loss is a primary focus throughout this dissertation.

1.1.2. Device Controllers

Device controllers provide the interface between input and output devices and the computer system. Although the capabilities of controllers vary widely, the basic functions of the controller are to allow software control of the device (*e.g.*, to start and stop the device) and to provide a data path between the device and the computer memory. Some controllers manage a single device, while others control many devices. Some controllers provide both audio and video support while others support only audio or video. For example, the SoundBlaster/16 from Creative Labs is a common sound card available for personal computers. The SoundBlaster supports both microphones and CD-players as input devices and speakers as output devices [22]. The SoundBlaster does not support video. The IBM/Intel ActionMedia I card, on the other hand, supports not only audio input and output devices, but also video camera input and output to a video display [41].

For audio and video input devices, the controller periodically measures the value of the analog input signal, converts the signal into a digital representation, and makes that value available to the computer. This process is called *sampling* and the binary representation of the input signal is called a *sample*. Controllers for output devices reverse the process and convert digital signals to analog before transferring data to the output device.

Individual samples are grouped into *media data units*. For video, the media data unit is usually a collection of samples representing all the pixels in a still image (*i.e.*, the video image at a particular sampling time). For audio, the media data unit is typically a set of audio samples collected over some interval. In this dissertation, the term *frame* refers to a media data unit for either audio and video. Input device controllers produce frames at a particular rate and output device controllers consume frames at the same rate. We refer to the process of acquiring a frame from the input device as

the *digitization* process and the process of playing the frame on the output device as the *display* process. Controllers are often *self-clocked* in the sense that timing generated by the controller itself, rather than the conferencing application or operating system, drives the generation or consumption of frames. For example, the frame rate for NTSC video, the standard for television in North America, is 30 frames per second [16] and video controllers supporting NTSC produce and consume video frames at this rate. With an NTSC input controller, a captured frame is available about every 33 milliseconds based on the clock of the input controller. Each captured frame records an image that is to be displayed for approximately 33 milliseconds. On an NTSC output controller, display of a captured image may be initiated only at specific points in time, called *display points*. The period between display points is also 33 milliseconds, but the actual time of the display points is based on the clock of the output controller and is not necessarily synchronized with the clock on the input controller that captured the image. The strict timing requirements of the input and output controllers has a significant impact on the design of the conferencing software and is discussed in more detail later.

1.1.3. Encoders and Decoders

Sampling audio and video streams produces a lot of data. A video stream with 256 × 240 pixel images, produced at NTSC video rates of 30 frames per second, and with 24 bits of color information per pixel has a data rate of approximately 44 megabits/second. In comparison to video, audio has a much lower bit rate, but in absolute terms, the audio bit rate is not small. CD-quality, stereo audio produces 2 audio channels with each channel sampled 44,100 times/second. There are 16 bits/sample, so the total bit rate is approximately 1.4 megabits/second. These data rates stress the capacity of most existing computer systems and data networks.

Fortunately, there is a large amount of redundancy in audio and video streams and people are very good at adapting to playback of incomplete or imperfect media. As a result, frames can often be heavily compressed using *lossy* compression techniques. With lossy compression, the decompressed binary data does not exactly match the original source binary data, but with audio and video the differences are often below the thresholds of human perception. The purpose of the audio and video encoders and decoders is to perform frame compression and decompression. Either hardware or software may be used to encode and decode frames and many device controllers

include these functions as part of the services of the controller. Typically, the same hardware or software can perform both the encoding and decoding of the frames and the term *codec* (*i.e.*, coder-decoder) refers to the component performing these functions.

Audio and video compression can achieve significant reductions in stream bit rates. For example, common compressed video formats such as H.261 and MPEG have bit rates between 64 kilobits/second and 6.0 megabits/second [9, 23, 36, 73, 113]. Common compressed audio formats such as PCM, DPCM, μ-law, and MPEG audio compression have bit rates between 4 and 64 kilobits/second per channel [11, 16, 83]. These data rates are significantly lower than the uncompressed rates. There are many media compression techniques and several are discussed in detail in Chapter 2.

1.1.4. Network Service

The network service consists of a network controller that physically connects the computer to a network, and software that allows applications to send and receive messages across the network. We assume the physical network is composed of a series of interconnected subnetworks. We call the devices that interconnect the subnetworks *routers*. We assume, unless otherwise stated, that the network technologies used for the subnetworks are network technologies in wide-spread use at the time of this dissertation. Typical examples of such technologies are local area network technologies such as Ethernet, token ring, and FDDI, and wide-area network technologies such as T1 and T3 leased lines [12]. Figure 1-2 shows a sample network. We briefly consider ATM networks in the last chapter.

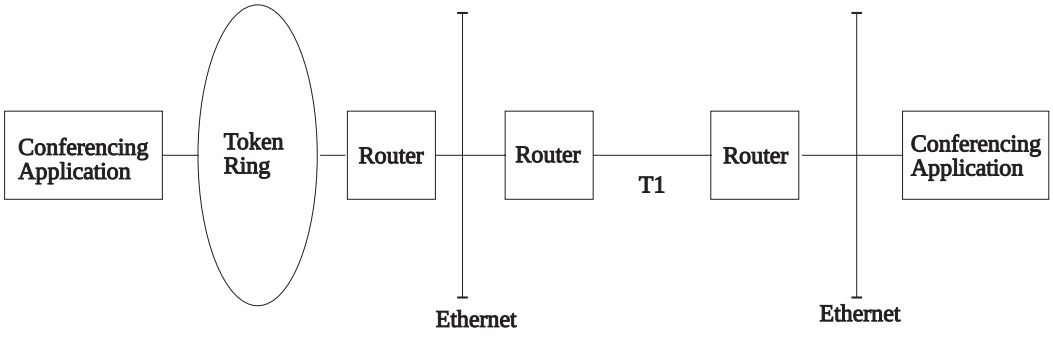


Figure 1-2: Sample Network

The network service provides a logical abstraction of the network that isolates the applications from the network technology and from the physical topology of the network. The combination of the network controller and the network software implement a set of *protocols*. A protocol defines the syntax, semantics, and ordering of data exchanged between two entities on the network [101]. The protocols describing a network architecture such as TCP/IP [20], SNA [38], or OSI [106] are arranged as *stacks* where each layer in the stack gives a different level of abstraction for a network communication [106]. We do not assume a particular network architecture for this dissertation (although the experiments described here use TCP/IP). We simply assume that the network interface provides a means of sending and receiving messages to a designated receiver. The application sends and receives messages by issuing commands to a *transport layer* that provides the interface layer between the application and the network service. The messages may be of arbitrary size and consist of one or more frames.

Since messages may be of arbitrary size, the network may not be able to carry a message in a single packet. As a result, the network service may fragment messages into packets for actual transmission through the network. The conferencing application is generally not aware of the number of packets generated for a given message or of the size of the packets. At a receiving station, the network service reassembles packets into messages before delivery to the receiving application. We assume the network service provides no guarantees about the delivery of a particular message (*i.e.*, it may be lost, delivered out of order, or corrupted) and there are no bounds on the time required to transmit a message from the sender to the receiver. We call such a network a *best-effort* network. We call networks that provide guarantees on the delivery and transmission times of messages *reservation-based* networks. We discuss both types of networks in more detail in Chapter 2, but our primary interest in this dissertation is on video conferencing over best-effort networks .

1.1.5. Conferencing Application

The conferencing application (1) provides a user interface that allows the person using the conferencing system to control the operation of the system; (2) establishes and breaks conference connections with other conferencing applications on remote

computers; and (3) controls the generation, transmission, and consumption of frames for each media stream.

1.1.5.1. User Interface

The user interface allows the user to control the operation of the application. The most obvious need for a user interface is to allow the user to identify with whom she wishes to conference; however, the conferencing system may provide a wide variety of other controls such as selecting the quality of the audio and video transmitted, setting the maximum transmitted bit rates, positioning video windows on the computer monitor, controlling the volume of audio, *etc.* This dissertation does not consider user interface issues and simply assumes the user has some mechanism for establishing connections and that any control over the formatting and presentation of the media (*e.g.*, screen positioning, volume control, *etc.*) does not affect the basic operation of the system. In particular, we assume the conferencing software has exclusive control over the selection of media coding schemes and the packaging of media frames into messages.

1.1.5.2. Establishing Conference Connections

Establishing conference connections implies the conferencing system must be able to resolve logical names (*e.g.*, the names of individuals or computers) into network addresses so that the conference media streams can be directed to the appropriate destination. We do not assume a connection-oriented protocol [12], but instead simply assume the conferencing application can somehow identify the destination network address of all conference participants and send data to those participants. In this dissertation, we are not concerned with how logical names are mapped to network addresses or how messages are routed in the network. We simply assume that the mechanisms exist as part of the network service. Unless otherwise stated, we assume video conferences involve only two participants on two computers and thus communication is point-to-point and not multicast.

1.1.5.3. Controlling Generation, Transmission, and Consumption of Frames

We use the term *continuous media* to refer to media streams (typically digital audio and video streams) that produce a series of discrete media frames that are intended to be played contiguously in time [16, 45]. A fundamental characteristic of continuous media streams is that successive frames are semantically related and there are timing

requirements that must be preserved between frames for acceptable display of the stream. For example, suppose a series of graphic images is being used for animation. The individual images in the series must be displayed in a specific order and with specific timing constraints in order to generate the illusion of motion. If these constraints are not met, the perceived quality of the animation is adversely affected. As a result, we consider the animation stream to be a continuous media stream. On the other hand, a series of unrelated graphic images displayed in sequence is not a continuous media stream because successive images are unrelated and there are no timing requirements between images. Many of the challenges involved in building a video conferencing system are directly related to managing continuous media streams.

Figure 1-3 shows the processing steps for one continuous media stream in a conferencing system. For simplicity, the figure shows the steps for acquisition and display on a single computer. Figure 1-3 shows only one stream, but there may be more than one stream (*e.g.*, audio and video) between two conference participants.

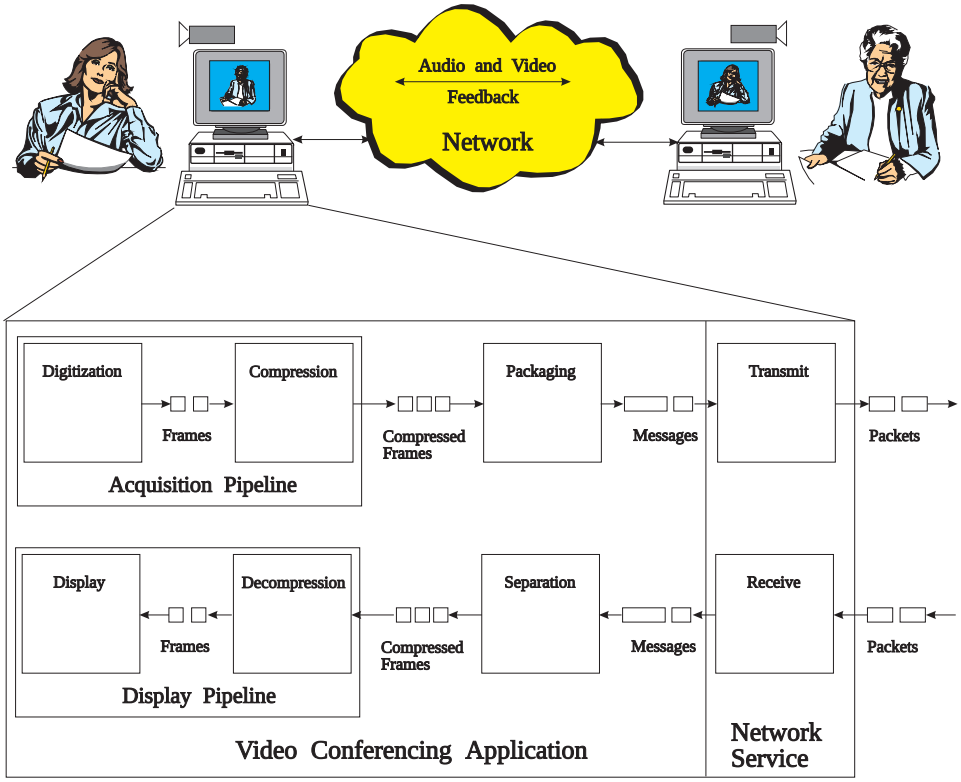


Figure 1-3: Video Conferencing Pipeline

We call each step in the sequence a *stage*. The sequence of stages forms a *pipeline* [61]. The stages of the pipeline are distributed across multiple computers. Each media frame must pass sequentially through each stage in the pipeline. Stages may operate concurrently (*e.g.*, a frame could be compressed while another is being digitized), but a particular frame may only be in one stage at a time. There may be several media streams between any pair of conference participants and each media stream may have a separate pipeline.

The video conferencing application is responsible for controlling the pipeline for each media stream involved in the conference. Controlling the pipeline means (1) issuing commands to control devices and functions, including commands to acquire and display frames; (2) handling the movement of frames between pipeline stages; (3) setting the control parameters that govern the operation of each pipeline stage, and (4) maintaining the timing requirements for generation and consumption of frames across all stages of the pipeline.

The *Digitization* stage (see Figure 1-3) captures frames from an input device and converts the frames into a digital representation. The input controller performs the actual digitization of the frame, but the conferencing application is responsible for recognizing the availability of new frames and moving the frames to the next stage in the pipeline. As illustrated in Figure 1-3, moving frames between stages is conceptually equivalent to moving the output from one stage to an input queue for the next stage.

The frame *period* is the amount of time it takes for the *Digitization* stage to produce a new frame. Different devices may have different frame periods and the frame period for a particular media stream defines the maximum frame rate of that stream. The digitization process is normally driven by a clock on the input controller. Because a hardware controller performs the digitization process, there is little variance in the delay between the availability of successive frames. From the perspective of the conferencing application, the input controller clock is an external clock where the ticks of the clock coincide with the availability of a new frame. We refer to this clock as the *digitization* or *acquisition clock*. For example, Figure 1-4 describes a video input device that produces frames at NTSC rates (*i.e.*, 30 frames per second). The frame period is approximately 33 milliseconds and each tick of the acquisition clock corresponds to the availability of a new digitized video frame.

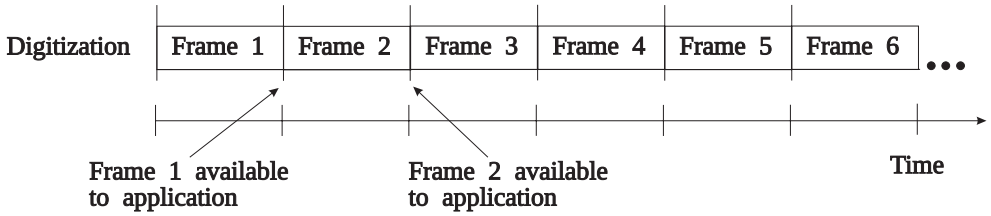


Figure 1-4: Timing Derived from Digitization Stage

The conferencing application must know when new digitized frames become available. Many input controllers generate an interrupt when a new frame becomes available, so each tick of the acquisition clock may correspond to an interrupt from the input controller. The operating system invokes the device driver associated with the controller in response to the interrupt and the device driver either directly passes the frame to the application or buffers the frame pending an application request for data. If the controller does not generate interrupts, the conferencing application must poll the device at precise rates to acquire frames. There are a limited number of buffers available to hold the digitized frames, so the conferencing application must manage the processing of the frames to guarantee the availability of an unused digitization buffer for each frame period.

The *Compression* stage compresses the digitized frame into some coded version of the original frame. The compression of a frame may begin immediately after digitization of the frame completes. The time needed to compress a frame may vary due to frame content, but the compression time must be less than the frame period so we can support the full frame rate from the digitization stage. The compression stage may have parameters that affect the speed, memory requirements, or degree of compression provided by the codec. Figure 1-5 shows the digitization and compression of a sequence of frames over time. The arrival of digitized frames implicitly clocks the start of the compression stage, so the time reference for digitization and compression is the same. Due to the relationship between the timings of the digitization and compression stages, we sometimes refer to the two stages together as the *Acquisition Pipeline* (see Figure 1-3). The acquisition pipeline is completely contained on the computer generating the media stream (*i.e.*, the conference source).

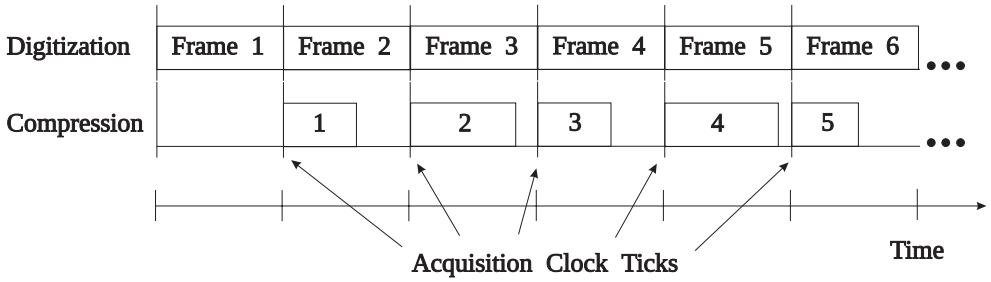


Figure 1-5: Digitization and Compression over Time

The *Packaging* stage collects some number of frames into a network message and periodically forwards messages to the *Transmit* stage for transmission across the network. The packaging stage is implemented in software and has parameters that control the number and type of frames packed into a message. The conferencing application may change these parameters at any point in the conference. We demonstrate later in the dissertation that controlling the packaging parameters in conjunction with the compression parameters is vital to acheiving satisfactory conferences.

Figure 1-3 shows the packaging stage as a part of the pipeline for a single media stream, but the packaging stage may receive input from several media streams and may emit messages composed of frames from multiple streams. The *Transmit* stage breaks the messages produced by the packaging stage into network packets and transmits the packets to the conference receiver. The *Receive* stage receives packets from the networks and reassembles the packets into a message. The network service performs the transmit and receive operations. These stages are outside the control of the conferencing application. The *Separation* stage is part of the conferencing application software and breaks messages into frames. If a single message carries frames from multiple streams, the separation stage places the frames on the appropriate decompression queue based on the frame type. We refer to the packaging, transmit, receive, and separation stages as the *Transmission Pipeline*. The transmission pipeline operates asynchronously from the digitization pipeline (*i.e.*, the digitization clock does not drive the transmission pipeline). The transmission pipeline is distributed across more than one computer. The packaging phase is contained on the conference source. The transmit stage may be entirely contained on the conference source or may consist of a series of transmit stages on multiple computers (*i.e.*, there may be many

intermediate nodes in the path). The receive and separate phases are on the conference destination.

The *Decompression* stage converts frames from compressed to uncompressed format. The *Display* stage takes an uncompressed frame from the decompression stage, converts the digital representation of the frame into a representation suitable for the output device (*e.g.*, analog signals for speakers), and plays the output on the device.

The output controller consumes data at a specific rate and the display stage must provide frames to the controller based on that rate. Video conferencing systems attempts to acquire and play media in real time, so for purposes of this dissertation the acquisition and play durations are the same[1]. For example, an audio frame captured over 16 milliseconds takes 16 milliseconds to play. Similarly, each frame of captured NTSC video represents approximately 33 milliseconds, so each frame should be displayed for approximately 33 milliseconds.

Although the period of the digitization and display stages are the same, the output controller has its own clock that defines the start of each display stage. Like the digitization clock, hardware drives the *display clock* and there is little variance in the clock period (the time between ticks). Like the digitization clock, the display clock appears to the video conferencing application as an external clock with a specific rate. Since different clocks drive the digitization and display stages, the start of each display stage is not necessarily synchronized with the digitization phase as illustrated in Figure 1-6.

The display clock may affect the decompression stage. In some video conferencing systems, the decompression stage can start immediately after the frame leaves the separation stage; however, with other systems, including the conferencing system used in this dissertation, the decompression stage must operate synchronized with the display clock. In these systems, frame decompression can only begin on a display clock tick, as illustrated in Figure 1-6, and the decompression time must be less than the display interval. Since the decompression and display stages may be synchronized, we refer to these two stages as the *Display Pipeline* (see Figure 1-3). The display

[1]When playing stored continuous media, the play duration is not necessarily the same as the capture duration (*e.g.*, video could be played in fast or slow motion).

pipeline is completely contained in the computer consuming the media stream (*i.e.*, the conference destination).

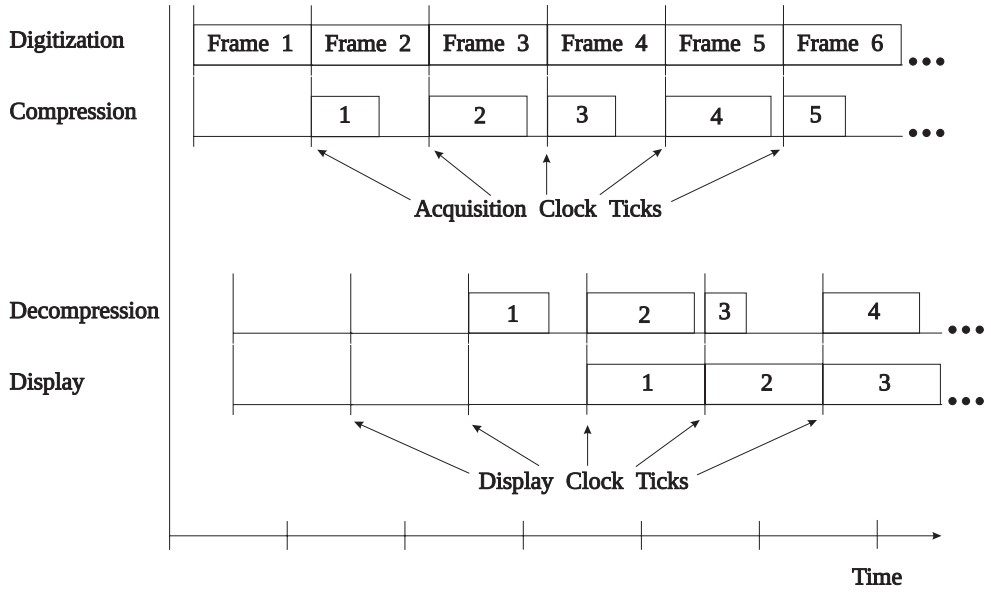


Figure 1-6: Digitization and Display Clocks

The acquisition and display clocks impose timing constraints on when the conferencing application must interact with the controllers. To operate correctly, the conferencing application must not only perform its functions properly, but must also meet the timing constraints imposed by the controllers. For example, to provide continuous play, a new frame must be available to the display stage at each tick of the display clock. If frames are not delivered to the display stage or are delivered after the display clock tick, there will be periods when no media is played. When an application must satisfy both functional and timing constraints to perform correctly, we say the application is a *real-time application.* Guaranteeing the performance of real-time applications requires careful construction of the system and application software using sophisticated techniques for CPU scheduling, management of data movement, synchronization, and general resource allocation and scheduling [3, 75, 57, 61, 91, 94, 103, 112]. In this dissertation, we assume the conferencing application and operating system are carefully constructed to guarantee performance of the acquisition and display pipelines, but that the network does not provide similar guarantees.

1.2 Network Congestion

Unfortunately, even if the acquisition and display pipelines are carefully managed, transmitting a video conference across best-effort networks can lead to conferences with poor quality. The network is a shared resource and many potential network consumers compete against one another for the relatively scarce network resources. When competition for resources is high, any given consumer may receive a share of the network resources that is inadequate to meet its needs. One way to solve this problem is to eliminate the competition by logically or physically dedicating resources to individual consumers. We call schemes that logically or physically dedicate network resources to individual consumers *resource reservation* schemes. Some researchers claim that this is the only way video conferencing can be successfully supported across data networks; however, dedicating network resources can be expensive, complicated, and usually requires changes to existing network components. For these reasons, we specifically focus on transmission of video conferences over best-effort networks, where network resources are not reserved. We claim we can usually deliver adequate quality conferences over best-effort networks using simpler algorithms than those used on resource-reservation schemes and without changes to the existing network components. Before discussing our scheme to accomplish this, we first discuss the nature and effects of network congestion.

From the perspective of a particular application, we say the network is *congested* if any message transmitted by the application, or more precisely any packet resulting from a message, waits for use of any resource while traveling from sender to receiver. The degree of network congestion is on a continuous scale, from a completely dedicated network path, where no packet ever waits for resources, to a completely blocked network, where all packets wait for resources forever. Congestion may occur whether the conference is entirely local (*i.e.*, within a "campus" network) or crosses wide-area networks. Local network conditions may cause significant congestion problems at the conference destination even for streams delivered "perfectly" (*e.g.*, using dedicated wide-area connections) up to the campus network [61].

The degree of congestion acceptable to a particular application varies with the application and the data streams involved. Congestion usually increases the delay in transmitting data from the data source to the data sink. Some applications can tolerate more delay than others. For example, a file transfer application may be very resistant to the effects of congestion because the acceptable elapsed time for a file transfer is

long compared with other applications such as real-time data acquisition. As long as all the file data arrives correctly within some relatively long period of time (*e.g.*, on the order of minutes), the file transfer is successful. In contrast, applications involving real-time data acquisition of sensor data can tolerate only very short network delays and may require dedicated channels to eliminate congestion as a potential source of delay. Video conferencing is a particularly interesting application in terms of congestion tolerance. On one hand, conferences have real-time delivery requirements because the display clock demands new frames be available on a regular periodic basis. Furthermore, human perception requires that the conference streams be delivered with low delay. These requirements limit the congestion tolerance of the conference. On the other hand, conferences also often have considerable flexibility in selecting the content of the transmitted data stream. The conferencing application is not required to transmit the entire media stream produced by the codec and may elect to transmit only a portion of the frames. The ability of the conference to reduce its network demands during congested periods enhances the conference's congestion tolerance. The challenge is determining how to change the network demands of the conference when network congestion occurs so that we maintain a quality conference.

There are two causes for network congestion: *capacity constraints* and *access constraints*. Capacity constraints limit the *bit rate* supported by the network. Figure 1-7 illustrates a capacity constraint. In this example, the computer labeled *Source* is attempting to send a video stream to the computer labeled *Sink*. The bit rate resulting from the compression stage is 2.0 megabits/second, but the capacity of the T1 line between routers *R1* and *R2* is only 1.544 megabits/second. Congestion occurs at *R1* because the incoming rate of 2.0 megabits/second is greater than the outgoing rate of 1.544 megabits/second. A queue of video data will form at *R1* and since the buffers at *R1* are limited, eventually some video data will be lost at *R1*. To address this constraint, the video source must reduce the bit rate transmitted to the sink.

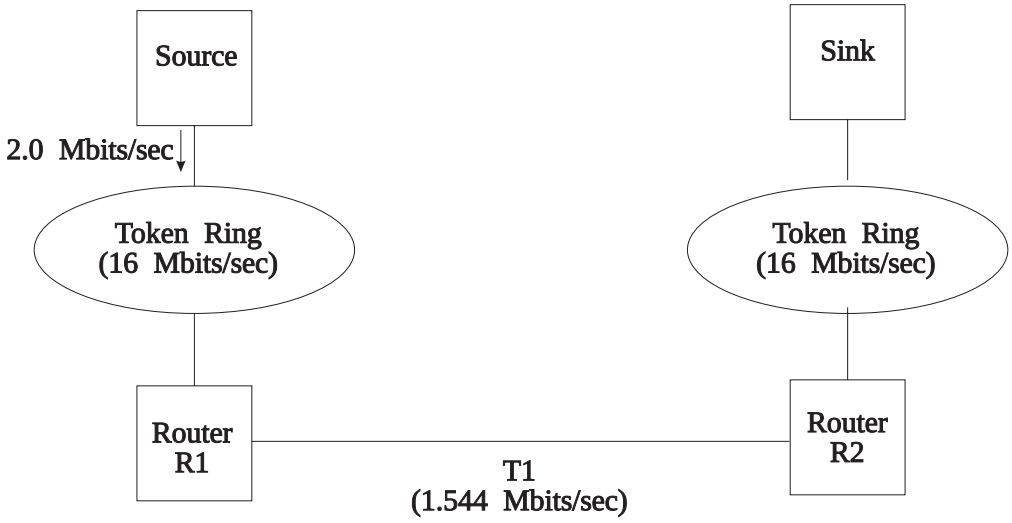


Figure 1-7: An Example of a Capacity Constraint

Access constraints limit the *packet rate* supported by the network. Figure 1-8 illustrates an access constraint. Again, the source is attempting to send a video stream to the sink. The source acquires video at NTSC rates and sends each video frame in a separate message (in this example we assume that each video frame fits in a single network packet for transmission). This means the source generates a packet about every 33 milliseconds. Assume in this example that several other computers are also attempting to transmit on the middle token ring (*e.g.*, the processors *P1*, *P2*, *P3*, and *P4* are contending for access to the shared medium). Due to competition for access to the middle token ring, the average wait time for tokens on the middle token ring (the time *R1* must wait for a free token to carry a packet) is 40 milliseconds. The packet rate from the source to *R1* is 30 packets per second, but the maximum outgoing rate from *R1* is only 25 packets per second. A queue of video data will form at *R1*, eventually the available buffers will fill, and again some video data will be lost at *R1*. To address the constraint, the source must reduce the packet rate transmitted to the sink.

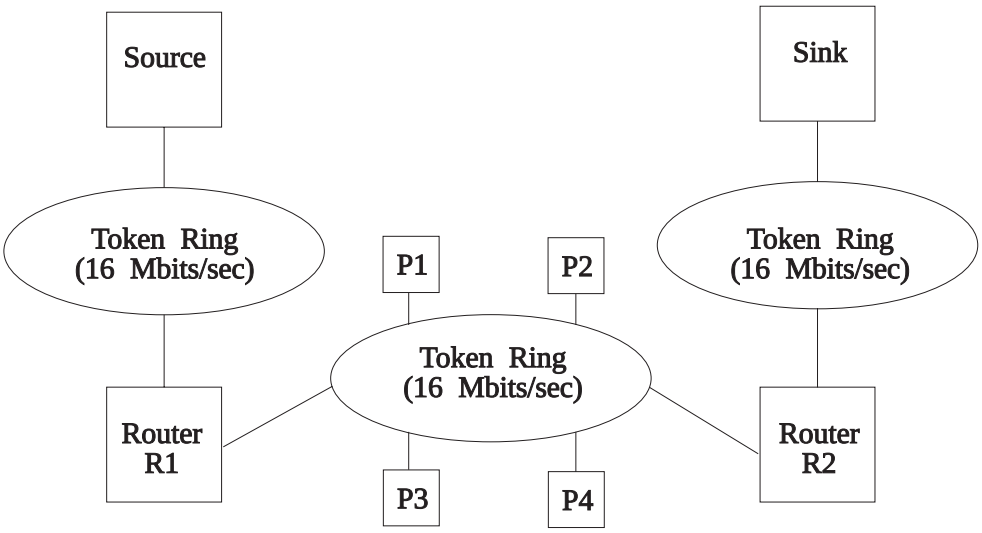


Figure 1-8: An Example of an Access Constraint

Capacity and access constraints can occur either separately or simultaneously and congestion resulting from these constraints lower the perceived quality of the conference by causing excessive latency, jitter, and frame loss in the media streams. We discuss each of these problems below.

1.2.1. Latency

The goal of desktop video conferencing is to provide computer-based communication that approximates a face-to-face conversation. A fundamental requirement is that there is a relatively short period of time between transmitting to someone and receiving their response. Unfortunately, network congestion can significantly increase the transmission delay of frames traveling through the network. When the delay or *latency* between acquiring the frame at the media source and playing the frame at the destination exceeds some "reasonable" time period, the performance of the video conference system does not support conversational interaction.

Several studies have attempted to quantify the "reasonable" latency bound for audio. Isaacs and Tang [51] observed a set of software developers using a video conference system to collaborate on a software development project and concluded that the maximum reasonable period for one-way audio latency is between 250 and 400 milliseconds [51]. The system users complained about system responsiveness even when audio delays were in the 300-400 millisecond range [51]. These results are

consistent with Wolf's [116] experiments that found audio latencies up to 420 milliseconds acceptable, but rated much lower in perceived quality than audio latencies of 167 milliseconds. Gruber and Le found one-way audio latencies of less than 40 milliseconds had no subjective effect on the perceived audio quality, delays of 300 milliseconds or less had little effect, and delays of 600 milliseconds or less had some effect, but were not found objectionable [42]. Various other researchers have suggested audio latency of 200 or 250 milliseconds are acceptable for video conferencing [10, 32, 95, 112]. There has been less work done on determining the "reasonable" latency bounds for video; however, video should be approximately synchronized with audio, so we claim similar bounds apply for video.

Regardless of the network conditions, some amount of latency is inevitable because of the delays associated with the media pipelines. For example, consider the pipeline for NTSC video shown in Figure 1-9. In this example, we assume the delay associated with transmitting the media frames across the network is zero. We also assume, for simplicity, that the acquisition and display pipelines are exactly synchronized. Given these assumptions, the delay between start of the digitization stage and start of the display stage for a given frame, the *display latency*, is about 99 milliseconds. Since the compression and decompression stages are synchronized with the acquisition and display stages, even if the compression and decompression stages each take less than 33 milliseconds, the delay remains 99 milliseconds.

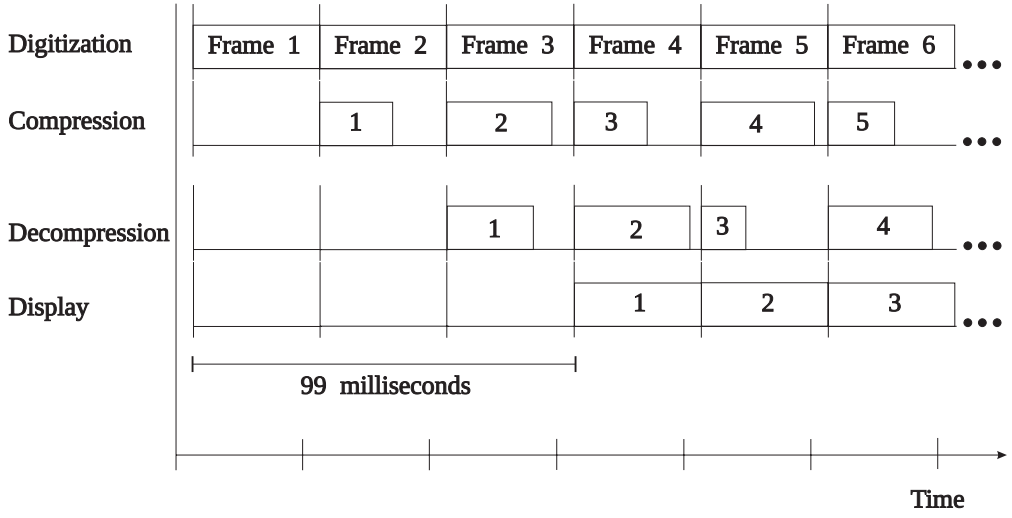


Figure 1-9: NTSC Video with Synchronized Generation and Consumption

In most conferencing systems, the acquisition and display clocks are not synchronized and a frame arriving at the destination computer may be forced to wait for the next tick of the display clock before entering the decompression stage. We call this wait the *synchronization delay*. The synchronization delay adds to the display latency of the stream. In the worst case, the synchronization delay may be equal to the display period. For example, in Figure 1-10 the synchronization delay for frame 1 is about 33 milliseconds and the resulting display latency is about 132 milliseconds.

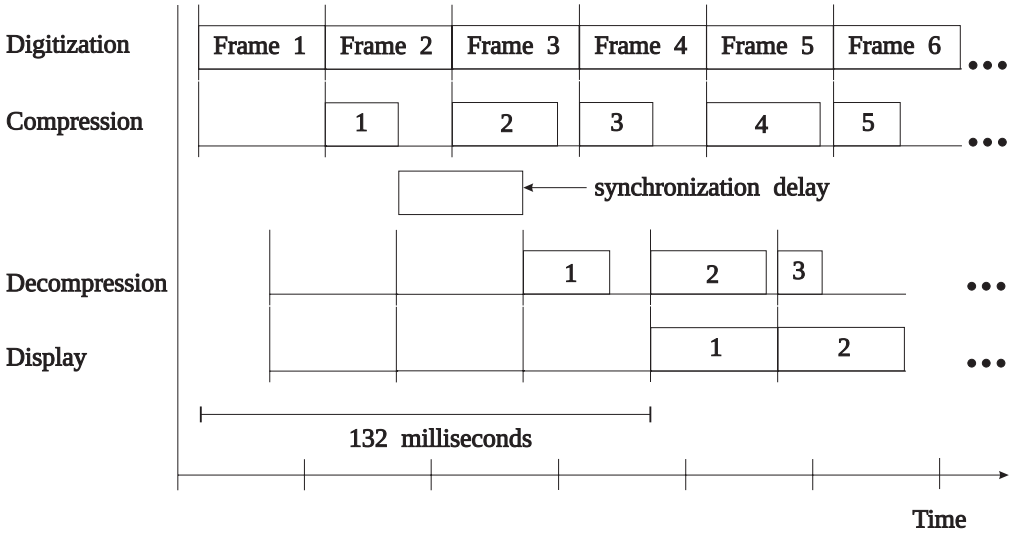


Figure 1-10: NTSC Video with Unsynchronized Generation and Consumption

Transmission of the frame may also add latency to the pipeline. For example, in Figure 1-11, the transmission time for each frame is about 25 milliseconds and the acquisition and display clocks are out of synchronization by 4 milliseconds. If the transmission time were zero, the first frame could be played with a display latency of about 103 milliseconds, but with the addition of the transmission delay, the actual display latency is 136 milliseconds. Depending on the synchronization delay, some additional latency may be unavoidable when transmitting frames across a network. Since the synchronization delay may be close to zero, we cannot eliminate the contribution of transmission delay to the display latency, but we can attempt to limit the transmission delay.

The latency associated with transmitting a frame across the network consists of the signaling time required to transmit the data across the physical medium for each

subnetwork plus the time required to gain control of the medium for each subnetwork and the delays (*e.g.*, packet processing time, queueing delays) experienced when passing through the routers connecting subnetworks. For a given number of bytes per frame, the physical signaling time does not change, but the time to acquire control of the subnetworks and the time spent waiting in router queues can vary because of network congestion. In this dissertation, we try to detect network congestion in the conferencing software and control the transmission of frames to limit the effects of congestion on the stream latency.

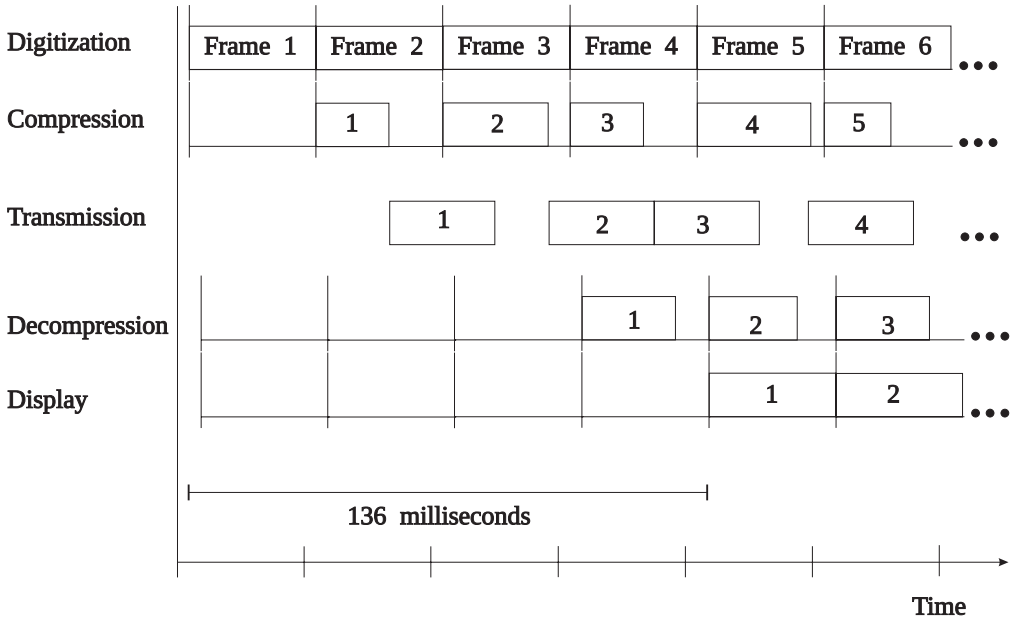


Figure 1-11: Frame Transmission Time Adds to Pipeline Latency

1.2.2. Jitter

Jitter is the variance in delay experienced by frames when passing through a sequence of one or more pipeline stages[2]. Jitter potentially disrupts the conferencing pipeline and introduces "gaps" while playing the media stream. Recall that the conferencing application must deliver frames to the display stage at the rate defined by the display clock. If no frame is available at a display clock tick, a *gap* occurs in the media

[2]Others often refer to this as "delay jitter" [32].

playback. A gap is an interval during which either nothing is played or the previous frame is replayed. The duration of the gap is the display period. The perceived effect of a gap depends on the media, the conferencing system, and the output devices. In the conferencing system used in this dissertation, audio gaps result in audible "pops." Video gaps result in the system replaying a previously played video frame.

To see how jitter can create gaps, consider the generation and consumption of video frames described in Figure 1-12. In this example, the transmission time for frame 1 is about 10 milliseconds and the display latency is 99 milliseconds. Frame 2 experiences a higher degree of congestion during transmission and takes about 30 milliseconds to cross the network. The display latency for frame 2 is 132 milliseconds. Because of the jitter between frames 1 and 2, there is nothing in the display queue when frame 1 completes playing and a gap appears in the video sequence.

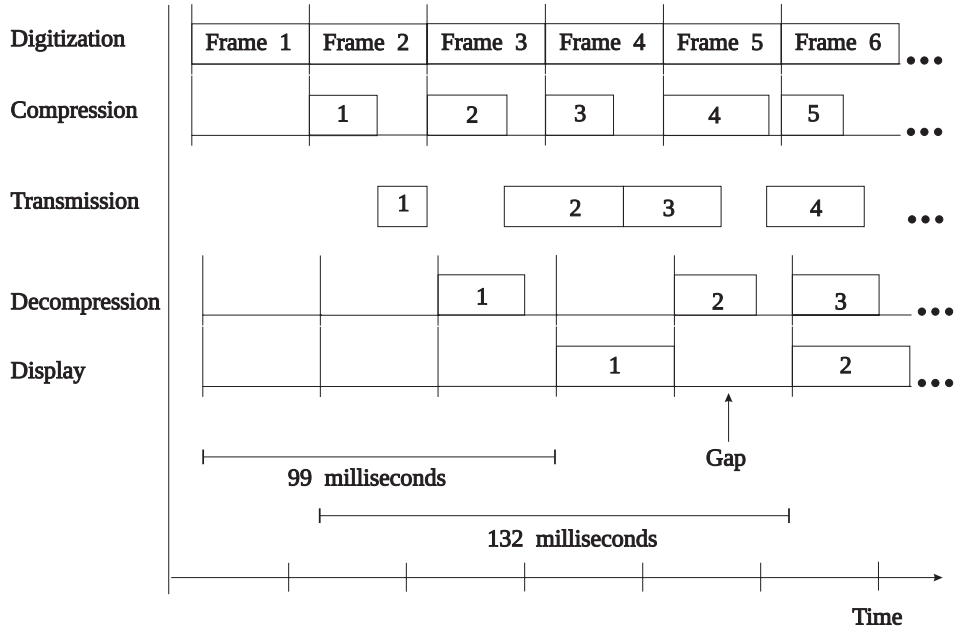


Figure 1-12: Jitter Resulting in Video Gap

Since gaps reduce the fidelity of the media stream, one of the primary responsibilities of the conferencing application is to limit the number and impact of gaps in the media streams. This implies that conferencing systems must control the effects of jitter in the frame delivery. Unfortunately, network congestion can greatly increase the amount of jitter present in a media stream. When the network is congested, queues of packets

build up at congested nodes and are released in a burst as congestion subsides. The delay between two consecutive frames delivered in separate bursts may be long while the delay between consecutive frames delivered in a single burst may be very short. This bursty delivery is quite common in best-effort networks and the interval between arriving frames at the conference receiver is likely to vary widely as a result.

Most conferencing systems compensate for jitter by using some frame buffering policy on the receiving side of the conference. The strategy used to manage the display buffer queue is called the *display policy* [103]. The basic idea is to buffer enough frames in the display queue so that even with the worst case jitter, new frames are guaranteed to be delivered to the receiver before the display buffer queue is emptied. In networks where the jitter between frames can be bounded and no frames are lost during transmission, display policies have been devised that guarantee no gaps occur [4]. Display policies can also improve the delivered frame rates in more general networks. For example, Stone investigated a number of display policies and found an adaptive queue monitoring policy often performs well in a general network environment [103]. However, if the jitter between frames is unbounded or frames can be lost during transmission, although buffering can reduce the number of gaps, buffering cannot completely prevent gaps. More importantly, buffering frames at the receiver directly adds to the display latency.

In this dissertation, we seek to limit the transmission jitter by carefully selecting the bit and message rates for each media stream. We do not explicitly consider display policies, but display policy techniques are complimentary to the techniques developed here. Although managing the transmission of the media streams can reduce network jitter, the conferencing application will likely still require some form of display policy.

1.2.3. Loss

When network congestion is severe, packets may be dropped because routers run out of available buffers. This packet loss leads to message loss since a message cannot be delivered if any of the packets comprising the message are lost. Message loss affects the fidelity of the video conferences. When a message is lost, all frames carried by the message are lost. Since lost frames cannot be displayed, gaps may appear in the played stream. Packet loss also implies the network is arbitrarily selecting which components of the media stream are delivered to the conference receiver. It is likely that the conference sender could make better choices about which parts of the media

stream to drop when only a portion of the stream can be delivered. From a network perspective, packet loss (or the resulting message loss) indicates that some component of the network is currently unable to support the aggregate demand for resources. The resources used to transmit a packet part of the way from source to sink are wasted if the packet is never delivered to the sink. According to Gerla and Kleinrock, this waste of network resources is the primary cause of the overall reduction in network throughput when congestion occurs [39]. Packet loss is thus bad from both the narrow perspective of the conferencing application and from the broader perspective of the network as a whole.

From a video conferencing perspective, there are a number of potential strategies for dealing with packet loss [112]. For example, it is sometimes possible to retransmit updates for portions of a frame (*e.g.*, a portion of the video image) to compensate for lost packets. Forward error correction schemes, such as redundant transmission of audio frames, may be able to reconstruct lost data or hide the effects of loss. Image and audio processing techniques such as interpolation or averaging of neighboring values can sometimes be used to mask the effects of loss. However, these techniques only limit the effects of packet loss. In this dissertation, we try to eliminate packet loss.

1.3. Conference Quality Measures

We claim that we can transmit conference streams over congested, best-effort networks and still preserve adequate conference quality. In order to demonstrate the validity of this claim, we now propose a set of measures for evaluating the effects of network latency, jitter, and loss on the quality of a conference. The purpose of these measures is to provide an objective set of criteria to compare the performance of different conferences.

Unfortunately, measuring conference quality is difficult. Many factors, such as the quality of the input and output devices, the coding and compression schemes used, the familiarity of the conference participants with each other, and the task to be accomplished, affect the perceived quality of the conference. Continuous media streams are often related and the actual quality of one media stream may affect the perceived quality of another (*e.g.*, anecdotal evidence says audio quality dramatically influences the user perception of constant video quality [63]). Many measures, including Mean Opinion Scores (MOS), signal-to-noise ratios (SNR), gap rates, and

playback latencies [10, 14, 32, 42, 51, 64, 86, 95, 103, 112, 116], have been used to measure conference quality, but no widely accepted comprehensive measure of quality has emerged. Determining overall conference quality is particularly complicated when component measures of quality conflict within a single measured conference or when trying to compare two conferences. The proposed measurements clearly do not cover all aspects of conference quality, but are adequate for purposes of this dissertation. In this dissertation, we use three measures (latency, fidelity, and loss) each of which is discussed in more detail below. A comprehensive measure of conference quality is clearly desirable, but is beyond the scope of this dissertation.

1.3.1. Latency Measures

We first consider a quality measure for conference latency. The display policy has a significant impact on the final display latency of the media frames since frame buffering may affect the display latency. However, we do not consider the display policy in this dissertation, and, for this reason, we distinguish between display latency and network latency. The *display latency* of a frame is the amount of time between when the frame entersthe digitization stage and when it enters the display stage. The *network latency* for a frame is the amount of the time between when the frame enters the digitization stage and when the frame leaves the separation stage on the receiving side (see Figure 1-3). We focus on network latency versus display latency since the transmission control policy can directly affect network latency, but ultimately the display policy controls the display latency.

Recall from our discussion of latency in section 1.21 that a number of researchers have proposed maximum latency bounds for conferencing latency. For this dissertation, we have adopted a 250 millisecond bound on network latency. Our bound is within the range considered acceptable by other researchers and has been used in a number of studies [10, 32, 42, 51, 95, 103, 112, 116]. We consider any media stream having network latency below 250 milliseconds as having acceptable quality from a latency perspective, although clearly lower latencies are preferred over higher. Below the 250 millisecond boundary, we consider two average network latencies within 50 milliseconds of each other to be indistinguishable. If two conferences have the same average network latency for a given media stream, we consider the conference with the lowest standard deviation for network latency (*i.e.*, the lowest network jitter) to have higher quality. We consider all periods where a media stream has network

latencies over 250 milliseconds to have unacceptable quality. When comparing two conferences to determine which conference has the best overall quality, we compare the average latency, the latency standard deviation, the frequency of periods with unacceptable latency, and the magnitude of latency "spikes" (periods of extreme positive jitter).

We also consider the latency differences between related streams. Some conferencing systems have separate acquisition and display pipelines for each media stream. The pipelines for different media streams may have a different number of stages and the duration of each stage may differ. This implies that data from different media streams may become available to the display stage at different times. For example, suppose the digitization period for both audio and video is 33 milliseconds, but compression of an audio frame takes 2 milliseconds and compression of a video frame takes 25 milliseconds. The compressed audio frame will be available for transmission 23 milliseconds before the compressed video frame. Suppose that due to the relative sizes of audio and video, it takes 1 millisecond to transmit the audio frame across the network, but 8 milliseconds to transmit the video frame. The audio and video frames are associated because they cover the same capture period and should be played together, but in this example the audio frame arrives at the destination 30 milliseconds before the corresponding video frame. If the streams are displayed independently, the audio frame may be played 30 milliseconds ahead of the corresponding video frame.

Maintaining the temporal relationship between frames from different media streams is called *media synchronization*. The most common example of synchronization requirements is for lip synchronization. The audio of people talking and the video of their lips moving to form the words should be played so that the two are roughly synchronized. The conferencing application is often responsible for determining which frames are temporally associated (particularly if different media streams originate from different controllers) and for playing temporally related frames within an acceptable range of synchronization. Strict media synchronization is not required because humans cannot detect slight differences in synchrony. Corresponding audio and video frames can be played within approximately 100 milliseconds of one another [102]. The interval is asymmetric in that audio can be played farther behind video than the reverse without perceptual impact [102]. Due to the difference in the speeds of sound and light, people are accustomed to sound reaching their ears slightly after the associated sight reaches their eyes because sound is slower than light. For example,

most people have experienced seeing the light from a fireworks display before hearing the fireworks explode. Jeffay, *et al*, have shown how audio and video synchronization can be exploited to improve conference quality [61].

This dissertation discusses and measures media synchronization, but only implicitly manages synchronization by controlling the transmission delay of the media streams. In this dissertation, we are more concerned with the overall latency of the streams than their synchronization (their relative latencies). We claim latency affects perceived conference quality more than synchronization. For example, Isaacs and Tang discovered that conference participants are more concerned with low audio latency than strict audio/video synchronization and that they prefer unsynchronized audio with latency in the 330 to 440 millisecond range to synchronized audio at 570 milliseconds [51]. For comparison purposes we consider corresponding audio and video frames to be adequately synchronized if they are played within 50 milliseconds of each other.

1.3.2. Fidelity Measures

In addition to conference latency, we would also like to evaluate the conference fidelity. Fidelity represents the information content of the conference. Due to the nature of human perception and continuous media, some degree of fidelity loss may be acceptable, but there are limits to the acceptable loss. Since the audio stream carries the majority of the data content of the conference, people are particularly sensitive to audio fidelity. We use gap rates to evaluate the combined effect of jitter and loss on audio fidelity. We have adopted the guidelines from Gruber and Le for acceptable audio gap rates [42]. Any gap greater than 10-15 milliseconds is audible. Good quality audio fidelity has fewer than 2.2 gaps per minute where no gap lasts more than 50 milliseconds. Fair quality audio has fewer than 5.4 gaps per minute with no gap over 50 milliseconds. We consider any gaps of greater than 50 milliseconds a severe violation of the quality constraints, since long gaps lead to unintelligible speech. These guidelines are consistent with limits suggested by Roussos, *et al* [95]. These guidelines are also consistent with informal experiments in our lab, where we have found that people can detect a gap of a single 16 millisecond audio sample when listening to music and 3-5 16 millisecond gaps per second is irritating and distracting. Loss of several consecutive 16 millisecond samples (*i.e.*, longer gaps) are more noticeable and objectionable than an equivalent number of non-consecutive losses spread over a period of time.

There is no authoritative estimate of the acceptable gap or frame rate for video. The speed of motion in the scene, the quality of the image in individual frames, application constraints, operating system or hardware architectures [94] and even audio quality [63] may affect the perceived quality of a sequence of video images. High frame rates can mask image quality problems. For example, Lippman claims the relatively poor image quality of NTSC television is acceptable only because of its high frame rate [74]. People are much more forgiving of video gaps than of audio gaps and the overall video frame rate appears to be a better measure of video fidelity than video gaps. Unfortunately, the estimates for acceptable frame rate varies widely between researchers. Many people claim video conferences should achieve NTSC frame rates (*i.e.*, 30 frames per second). Others claim much lower video frame rates are acceptable. For example, Wakeman suggests video frame rates as low as 2-3 frames per second (FPS) may be acceptable for video conferencing [112]. Isaacs and Tang studied a system with a default video frame rate of only 5 FPS [51]. Still others suggest rates somewhere in between (*e.g.*, Edwards suggests 15 FPS as an acceptable video frame rate [30]).

The experimental conferencing system used for this work is capable of generating VHS-like quality images at up to 30 FPS. In informal experiments with "talking heads" conferences, we have found people typically do not notice loss of 5 FPS out of 30 FPS. People may notice the difference between 30 and 15 FPS, but do not find 15 FPS particularly objectionable. Video frame rates below 10 FPS are quite noticeable and people do not consider frame rates below 5 FPS to be interactive. From this we have adopted the following guidelines for video fidelity: (1) video frame rates greater than 15 FPS are acceptable for video conferencing, with higher rates obviously preferred over lower; (2) two video frame rates between 15 and 30 FPS are generally indistinguishable if within 5 FPS of each other; (3) perceived video quality degrades rapidly below 15 FPS and should be avoided if possible; and (4) rates below 5 FPS are unacceptable.

1.3.3. Loss Measures

We also measure the number of lost frames and messages. Ideally, there is no loss. We want to avoid loss both from an application perspective, since loss can lead to gaps and lower played frame rates, and also from the network perspective since lost messages represent wasted network resources. From a perceptual perspective, we do

not propose an acceptable level of loss since the loss requirement is implied by the acceptable frame rates and number of gaps. That is, loss must be such that the fidelity requirements described in section 1.3.2 are satisfied. From a network perspective, we want the conferencing application to match its message and bit rates to that sustainable by the network. From this perspective, our goal is no loss.

1.3.4. Summary of Quality Measures

To summarize, in this dissertation video conference quality is measured by the network latency of the audio and video streams, the delivered frame rates for the audio and video streams, the number of gaps experienced by each media stream, and the number of packets lost during the course of the conference. These measures clearly do not cover all aspects of conference quality and the relatively simple counts and averages adopted here do not completely cover the statistical nature of perceptual quality. Nevertheless, all the proposed measures are objective, concrete, and have a clear relationship to perceived quality.[3]

	Video	Audio
Network Latency	0-50 *ms* - excellent 51-100 *ms* - very good 101-150 *ms* - good 151-200 *ms* - fair 201-250 *ms* - poor > 250 *ms* - unacceptable	0-50 *ms* - excellent 51-100 *ms* - very good 101-150 *ms* - good 151-200 *ms* - fair 201-250 *ms* - poor > 250 *ms* - unacceptable
Fidelity	26-30 FPS - excellent 21-25 FPS - good 15-20 FPS - fair 11-14 FPS - acceptable 5-10 FPS - poor < 5 FPS - unacceptable	≤ 2.4 gaps/minute - good ≤ 5.4 gaps/minute - fair > 5.4 gaps/minute - poor No gaps > 50 milliseconds
Loss	0 packets	0 packets

Table 1-1: Summary of Quality Measurements

In the experiments in later chapters, it is generally the case that the different transmission schemes produce consistent results across all the categories of quality

[3] We could also have measured the quality of the conference in terms of the relative perceptual quality achieved by different coding schemes (*e.g.*, the relative quality of stereo audio to monaural audio), but we chose not to do this.

measures. For example, if scheme A delivers conferences with significantly lower latency than scheme B, scheme A also usually delivers a conference with fewer gaps than that with scheme B. Occasionally, there are conflicting results across quality measurements and these cases are dealt with on a case-by-case basis.

1.4. A Transmission Control Framework

The focus of this dissertation is on managing the transmission pipeline of the video conference. We call a strategy or algorithm for managing the transmission pipeline a *transmission control policy* or *transmission control scheme*. The transmission control policy is implemented in the conferencing application and is independent of the network service (in particular, the transmission control policies discussed here have no relationship to the *Transmission Control Protocol* in the TCP/IP protocol suite [20]). The goal of the transmission control policy is to preserve acceptable conference latency and fidelity even if network performance is degraded by congestion. The transmission control policy attempts to accomplish this goal by controlling how the media streams are generated, compressed, and transmitted.

The transmisssion control policy can control the media streams in two ways. First, the transmission control policy can adjust the parameters to the acquisition and compression stages to increase or decrease the generated bit rate of a media stream (*e.g.*, by changing the frame rate generated by the acquisition stage or the compression algorithm in the compression stage). Second, the transmission control policy can adjust the parameters of the packaging stage to control the number and types of media frames carried by a network message. We refer to changes in the acquisition and compression parameters as *scaling*. We refer to changes in the packaging parameters as *packaging*. Packaging and scaling are complementary techniques. Scaling is often effective when the network is capacity constrained and packaging is often effective for access constraints. Figure 1-13 shows the transmission control policy in the context of the video conferencing pipeline. In the figure, the arrows from the *Transmission Control Policy* box to the *Digitization*, *Compression*, and *Packaging* boxes indicate that the transmission control policy sets the parameters that govern the operation of the digitization, compression, and packaging stages in the pipeline. Figure 1-13 also shows that the display policy controls the display and decompression stages. This figure illustrates that the transmission control policy and the display policy are

complementary techniques with each controlling a different part of the conferencing pipeline.

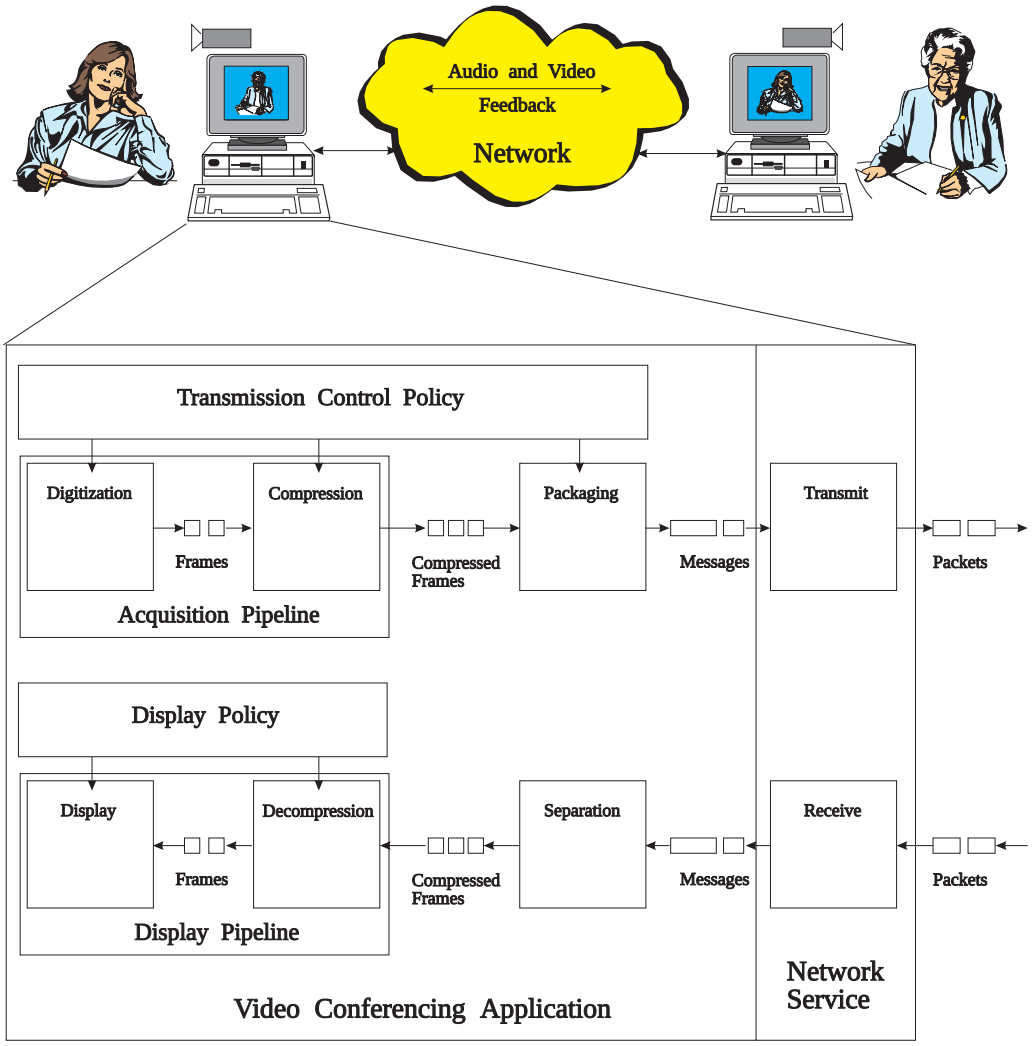


Figure 1-13: Transmission Control Policy and Display Policy in Conferencing Pipeline

Several video conferencing systems have been built that dynamically adjust to network congestion in a best-effort network. Scaling has been the primary technique used to address congestion in these systems. The basic scaling strategy is to reduce the transmitted bit rate when congestion is high and to increase the bit rate when congestion is low. The idea is to reduce the network demands of the conference, in terms of bit rate, when congestion is high, then improve the conference fidelity by increasing the media bit rate when congestion is low. Scaling is usually applied to the

video stream since video typically has much higher bit rates than audio. In some cases, particularly in networks that are capacity constrained, video scaling has proven an effective technique for dealing with congestion [14, 18, 64]; however, we demonstrate later that scaling is less effective than packaging when dealing with congestion caused by access constraints.

The basic packaging strategy is to decrease the number of messages (*e.g.*, by increasing the number of frames carried by a message) when congestion is high and increasing the message rate (*e.g.*, by decreasing the number of frames carried in a message) when network congestion is low. The idea is similar to scaling in that packaging attempts to reduce the network demands of the conference when congestion is high and improve the quality of the conference when congestion is low. Packaging differs from scaling in that when congestion is high, packaging reduces contention for network resources in terms of packet rate rather than bit rate. Also, when attempting to increase conference quality as congestion subsides, packaging attempts to lower conference latency while scaling attempts to improve conference fidelity. Packaging is related to scaling in the sense that large increases or decreases in the stream bit rate may indirectly force changes in the stream packaging. Indeed, we claim the changes to packaging resulting from large bit rate changes explains some of the success of video scaling techniques. Unfortunately, the value of packaging is often not recognized and most existing video conferencing systems do not exploit packaging in their adaptations. Our thesis is that an end-to-end transmission control scheme that uses both scaling and packaging can sustain a low-latency, high-fidelity conference over best-effort networks even when the network is heavily congested.

There are two issues to resolve when adapting to network congestion. First, we must determine whether scaling or packaging is the best strategy to address congestion at a particular point in the conference. Second, we must assess the impact of scaling and packaging on conference fidelity and latency. For example, reducing the bit rate of a stream through scaling may reduce the fidelity of the played media (*e.g.*, using a video compression scheme with better compression, but poorer image quality). Decreasing the number of messages may increase the latency of the media stream (*e.g.*, holding frames at the conference source until a number of frames are available for transmission rather than transmitting the frames as they become available).

We address these two issues by developing a transmission control framework that (1) describes the capabilities of a video conferencing system as a set of *operating points*,

(2) shows the impact of access and capacity constraints on each operating point, (3) illustrates how changes in network congestion changes the set of operating points that may be used at any point in the conference, and (4) relates the operating points and network congestion to the perceived quality of the conference.

The selection of a particular set of parameters for the acquisition, compression, and packaging stages of the pipeline defines an *operating point* for a media stream. A conferencing system can only use one operating point at a time for each media stream, but is free to change operating points over time. There are a finite set of combinations of parameters, so there are a finite number of operating points for any conference stream. The set of operating points for a media stream describes all possible adaptations that the transmission control policy can make to that stream in response to changing network conditions. Since operating points define the generation and compression parameters as well as the message rate, each operating point can be described by a distinct combination of bit rate and message rate. Using these rates, the framework describes the effect of capacity and access constraints on the desirability of particular operating points. For example, if the network is capacity constrained, the conferencing system should not select an operating point with a high bit rate. The framework also describes the dynamic nature of network congestion and how a particular operating point may sometimes be a desirable operating point and other times not, depending on the level of network congestion and the type of network constraint. Finally, the framework describes the effect each operating point has on the perceived quality of the conference and thus provides guidelines and heuristics for adapting to network congestion such that the quality of the conference is preserved.

We have developed a transmission control policy, called *Recent Success*, based on the transmission control framework. The *Recent Success* algorithm dynamically adapts the transmission of the continuous media streams to current network conditions by both scaling and packaging the streams. We empirically demonstrate, using both controlled network experiments and production network experiments, that the algorithm can deliver low-latency, high-fidelity conferences over networks that are capacity constrained, access constrained, or both. We show the algorithm works over a variety of network topologies and technologies, including networks where messages are fragmented during transmission. We show that adaptation of both the bit and message rates produces conferences with lower latency and higher fidelity than those

produced using non-adaptive transmission algorithms or by those produced by scaling the video bit rate alone.

1.5. Overview of Dissertation

The rest of this dissertation is organized as follows. Chapter 2 surveys several media coding and compression schemes, a number of media scaling techniques, and the transmission control schemes used by several existing video conferencing systems. Chapter 3 introduces the complete transmission control framework. Chapter 3 also discusses the nature of network congestion and the impact of capacity and access constraints. We demonstrate that relieving capacity and access constraints requires fundamentally different adaptations to the media streams. Chapter 4 describes the *Recent Success* algorithm and gives an implementation of the algorithm in the C programming language. Chapters 5 and 6 use the algorithm implementation described in Chapter 4 in a series of controlled network experiments (Chapter 5) and production network experiments (Chapter 6). Chapter 7 discusses our conclusions.

Chapter II
Related Work

This chapter introduces some of the concepts, terminology, and related work pertinent to the rest of the dissertation. It provides an overview of audio and video coding techniques, a discussion of media scaling techniques, and a brief survey of the transmission control strategies used by commercial or research video conferencing systems. It does not give an exhaustive survey of any of these topics or describe the details of the techniques or algorithms, but instead provides a general overview of the most common or widely known algorithms.

2.1. Survey of Audio/Video Coding Techniques

2.1.1. Audio Coding

2.1.1.1. Pulse Code Modulation (PCM)

Sound is produced when a vibrating object (*e.g.*, the strings on a guitar) compresses air molecules (pushes the air molecules into a smaller space), generating areas with high pressure (high density of air molecules) and low pressure (low density of air molecules). *Sound waves* are the alternation of high pressure regions with low pressure regions [6]. The higher the air pressure (the higher the *amplitude* of the sound wave), the louder the sound.

Sound waves are continuous; the air pressure at a point in space varies continually. Representing a sound wave in digital form requires sampling the sound wave at periodic intervals and assigning one of a finite set of values to represent the amplitude of each sample [16]. The process of sampling a continuous signal and assigning discrete values to the samples is called *Pulse Code Modulation (PCM)* and is illustrated in Figure 2-1.

We can measure the *frequency* of the sound wave by counting the number of "crests" (local maximas of the air pressure) of the sound wave at a particular point in space

over an interval [6]. Frequency is measured in Hertz (Hz). One Hertz is one crest or *cycle* per second. To ensure accurate sampling of the signal, the sampling rate must be at least twice the maximum signal frequency [83]. For example, the human voice has a range of approximately 4 kHz, so a sampling rate of 8 kHz is needed to accurately capture the voice signal. Humans can hear approximately a 20 kHz range of frequencies so sampling frequencies of 44.1 kHz and 48 kHz have been adopted as standard sampling rates for digital equipment [16].

The process of assigning the sample a discrete digital value is called *quantization* [11]. The quantized values represent the amplitude, or loudness, of the audio signal. Since digital values are discrete and sound waves are continuous, a range of amplitudes must be mapped to each digital value. The difference between the actual amplitude and the digital representation of the amplitude is called the *quantization error* [11]. Naturally, the more bits used to represent the sample, the less the potential quantization error. Eight bit and sixteen bit values are typically used for quantization. *Quantization noise* is the difference between the digital reproduction and the original continuous signal. Eight bit values have an approximate range of 48 decibels [83]. The quantization noise with 8 bit quantization is audible. Sixteen bit quantization is significantly better than 8 bit, with a range of approximately 96 decibels [83], but is still not capable of representing the full range of human loudness perception (approximately 120 decibels) [16].

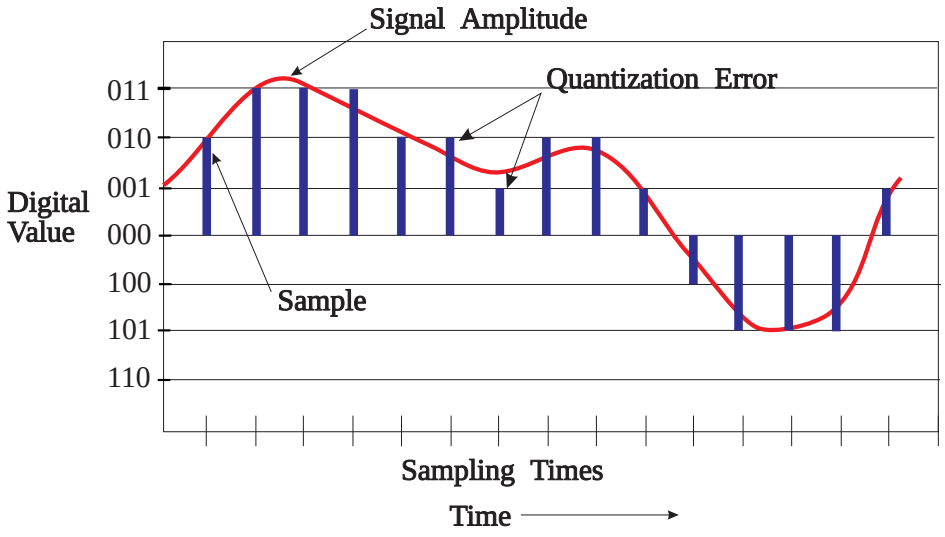


Figure 2-1: Pulse Code Modulation (PCM) (adapted from [11])

2.1.1.2. Differential PCM (DPCM)

The amount of data generated by an audio stream is small in relation to that of a video stream, but the amount of data in an audio stream is not small in absolute terms. For example, an audio CD produces 2 channels of audio sampled at 44.1 kHz using 16 bit quantization levels. This corresponds to a PCM bit rate of 1.4 megabits/second (*i.e.*, 44,100 samples/second × 16 bits/sample × 2 channels) [83]. The high bit rate of audio has led to the development of compression schemes to reduce the size of the audio stream. Differential Pulse Code Modulation (DPCM) and Delta Modulation (DM) are two techniques that have been used to compress a set of digitally sampled audio values. Both of these schemes achieve reduction in the stream bit rate by encoding successive samples as differences between the current sample and some predicted sample.

Figure 2-2 illustrates the DPCM process. A quantized sample, $X(n)$, is fed into the audio encoder. The predicted sample, $Xp(n-1)$, is subtracted from the current sample and only the difference, $C(n)$, is recorded. Since the magnitude of the difference is likely much smaller than the actual quantized sample, the difference can be represented using fewer bits (*e.g.*, the original sample may be 16 bits, but the difference represented as 4 bits). Since the actual magnitude of the difference may exceed the encoded value (due to the limited number of bits used to represent the difference), the predicted value must be adjusted to correct for any induced errors. This is done by adding the old $Xp(n-1)$ value to the encoded difference $C(n)$ to produce the new prediction $Xp(n)$. The decoder reconstructs the sample by adding the encoded difference to the previously played sample.

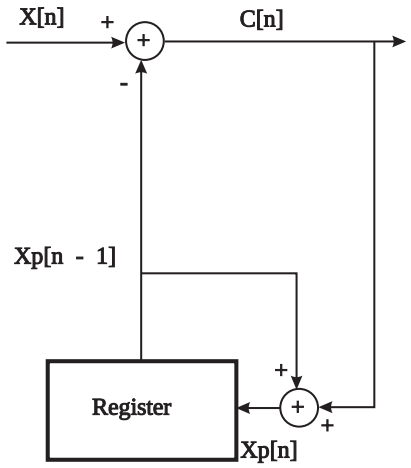


Figure 2-2: Differential PCM (DPCM) (adapted from [11])

Delta Modulation is very similar to DPCM, but the difference is encoded using only a single bit. For example, DPCM might use four bits to represent the difference between two successive 16 bit samples. DM would represent the difference with only a single bit. Due to the limited size of the difference (and thus large potential error), DM requires a high sampling rate to be reasonably accurate. The lower number of bits per difference with DM compared to DPCM is somewhat offset by the higher sampling rate of DM.

The compression ratios achieved by DPCM and DM are good (*e.g.*, 4 to 1 when 16 bit samples are represented by 4 bit differences), but because of the limited number of difference bits, both DPCM and DM suffer from a problem called *slope overload* [11] when the values of successive samples differ by more than the maximum difference that can be represented with the difference bits. As a result, streams with large differences between samples have large errors and it takes a period of relatively small changes to correct the error.

From a transmission control perspective, DPCM and DM are not loss tolerant. The decoder must see the same stream of differences as the encoder to accurately decode the audio stream. If differences are lost during transmission, the played audio has poor fidelity. One way to control the effect of loss in the stream is to periodically send the complete sample from the audio source to the receiver. The decoder can then periodically synchronize with the encoder and limit the effect of lost differences.

2.1.1.3. Adaptive DPCM (ADPCM)

Adaptive DPCM (ADPCM) attempts to resolve some of the problems caused by slope overload by changing the *step size* of the quantizer [83]. *Step size* is the granularity of the change between successive decoded samples. In a DPCM scheme, the implied step size is 1 and the new prediction is produced by adding the difference between the old prediction and the actual sample to the old prediction. For example, if the previous prediction is 90 and the current sample is 95, the difference between the two is multiplied by the step size of 1 and added to the previous prediction to calculate the new prediction (*i.e.*, new prediction = 90 + (100 – 95) × 1 = 95). There is no error in the calculation of the new prediction unless the difference exceeds the amount that can be represented in the difference bits. Suppose the DPCM scheme uses 4 bits to represent differences. Using a sign-magnitude encoding, the range of potential differences is (–7) to 7. Suppose that the previous sample is 90 and the new sample is 100. In this case, the difference between the two samples cannot be accurately represented and must be estimated as 7. The resulting prediction is 90 + (7 × 1) = 97 which does not accurately represent the sample of 100. ADPCM attempts to limit the error resulting from this slope overload problem by changing the step size based on the magnitude of the difference between the prediction and the sample. For example, when the prediction is 90 and the sample is 100, if the step size is changed to 2 rather than 1, the difference could be coded as 5 and the new prediction has no error since 90 + (5 × 2) = 100.

Figure 2-3 shows a diagram for an ADPCM encoder [83]. $D(n)$ is the difference between the sample, $X(n)$, and the prediction, $Xp(n–1)$. $D(n)$ is the input to a quantizer algorithm. The output of the quantizer, $C(n)$, is a coded value representing a multiplier of the current step size. The decoder reconstructs the sample by decoding $C(n)$ (via the dequantizer), multiplying the current step size by the decoded multiplier, and adding this result to the current predictor. The encoder follows a similar strategy to adjust its predictor. Potential step sizes are typically held in a table. The current step size is identified by an index into the step size table and the coded value $C(n)$ is used with a second table to determine the adjustment to be made to the step size table index, which effectively changes the step size. Both encoder and decoder use the same step size table and step size adjustment table. The encoder and decoder stay synchronized as long as the receiver sees the entire series of coded values produced by the sender.

The advantage of the ADPCM scheme over DPCM and DM is that the variable step sizes limit the slope overload problem. Large differences between the incoming sample and the prediction lead to coded values, $C(n)$, representing large multipliers to the current step size. Large differences also increase the step size, so rapid changes in sample magnitudes can be quickly accommodated. Due to the large step sizes and corresponding lack of granularity on the decoded differences, the decoded sample may not exactly match the source sample, but the error is less than with DPCM or DM. Stable periods, where differences between samples are small, eventually lead to small step sizes. Smaller step sizes allow finer adjustments to the predicted sample, improving the accuracy of the decoded sample. The ADPCM scheme has the same problems as DPCM and DM when differences (or coded differences in the case of ADPCM) are lost during transmission.

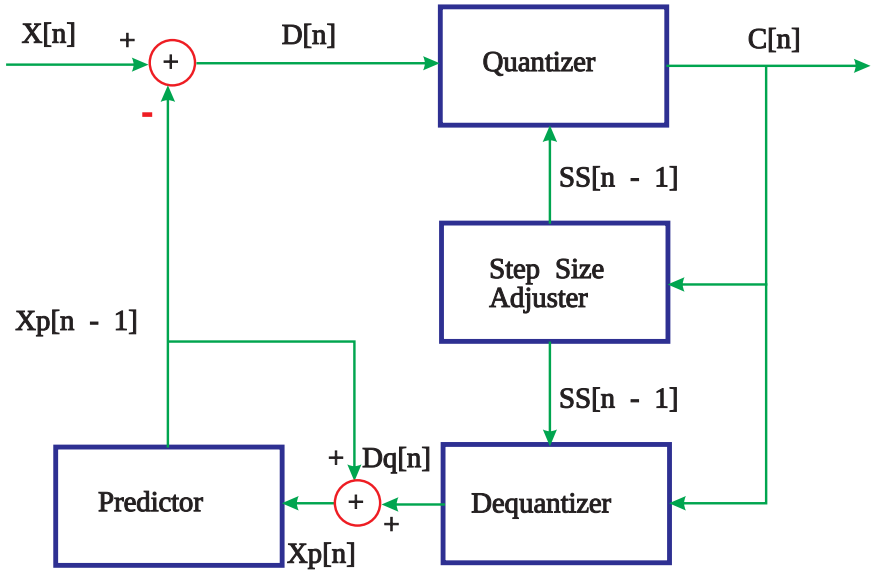


Figure 2-3: Adaptive Differential PCM (ADPCM) (adapted from [83])

2.1.1.4. μ-Law Compression

The DPCM, DM, and ADPCM schemes all seek to exploit the temporal redundancy of the audio streams, since it is likely that audio samples taken close together in time will vary only be relatively small amounts. The μ-law compression algorithms take a different approach. A logarithmic transformation allows a given number of bits to cover an extended range of values (*e.g.*, 8 bits used to cover a 14 bit range of values)

[83]. This expansion in range is achieved at the expense of accuracy. Due to the logarithmic scale, μ-law encodes low amplitude values more accurately than high amplitude values. This loss of accuracy is acceptable only because people are more sensitive to differences in low amplitude values than to high amplitude [83].

Conceptually, a sample is encoded by first scaling the sample value to a value between (–1) and 1. The resulting value, x, is used in the following equation to calculate the encoded value $F(x)$ [11].

$$F(x) = sign(x) \times 127 \times \ln(1 + \mu|x|) / \ln(1+ \mu)$$

In practice, the values for $F(x)$ are usually pre-computed and stored in a table for fast compression and decompression. Table 2-1 taken from [11] shows the encoding for table for $\mu = 255$. This encoding is used in North America and Japan for T1 digital telephone service [83]. Note that the step size within a table row increases with higher input amplitudes, with a corresponding decrease in decoding accuracy.

The μ-law encoding scheme is attractive from a transmission control perspective because the receiver can decode each transmitted sample independently, so the stream is more tolerant of loss than the difference coding methods. Furthermore, μ-law encoding and decoding are simply table lookups, so encoding and decoding are fast and easily implemented. For example, we encode an input amplitude of 500 by using 500 as an index into an encoding table to find the coded value of 64 (see Table 2-1). To decode, we use the coded value of 64 as an index into a decoding table to find the decoded value of 495. Although encoding and decoding are fast, μ-law encoding does not achieve the compression ratios possible with the differencing methods, so the resulting audio bit rate is higher. For example, it takes 8 bits per sample to represent the sign and magnitude of the coded values in Table 2-1, but an ADPCM encoding may use only 4 bits per sample to encode the same audio stream.

Input Amplitude	Step Size	Segment	Quantization	Code value	Decode Amplitude
0-1	1		0000	0	0
1-3 3-5 ... 29-31	2	000	0001 0010 ... 1111	1 2 ... 15	2 4 ... 30
31-35 ... 91-95	4	001	0000 ... 1111	16 ... 31	33 .. 93
95-103 ... 215-223	8	010	0000 ... 1111	32 ... 47	99 ... 219
223-239 ... 463-479	16	011	0000 ... 1111	48 ... 63	231 ... 471
479-511 ... 959-991	32	100	0000 ... 1111	64 ... 79	495 ... 975
991-1055 ... 1951-2015	64	101	0000 ... 1111	80 ... 95	1023 ... 1983
2015-2143 ... 3935-4063	128	110	0000 ... 1111	96 ... 111	2079 ... 3999
4063-4319 ... 7903-8159	256	111	0000 ... 1111	112 ... 127	4191 ... 8031

Table 2-1: μ255 Encoding/Decoding Table

2.1.1.5. MPEG Audio

The Motion Picture Experts Group (MPEG) has taken a much more sophisticated approach to audio compression than the techniques discussed so far. The basic idea behind MPEG audio compression is to exploit the characteristics of human perception to achieve high compression rates. Even with high levels of compression, MPEG audio can be perceptually lossless (*i.e.*, expert listeners cannot distinguish between the source audio and audio produced from the compressed representation). In evaluation tests, MPEG audio compression reduced a stereo audio stream sampled at 48 kHz with 16 bit samples, with a raw bit rate of approximately 1.54 megabits/second, to a 256 kilobits/second stream that expert listeners could not distinguish from the original

stream [83]. Other schemes, for example the differencing algorithms, can achieve the same level of compression, but the results are perceptually lossy. Unfortunately, the sophistication of the MPEG strategy also means the MPEG algorithms are complex and usually require special hardware to implement in real-time.

MPEG achieves the perceptually lossless results by exploiting a feature of human perception called *auditory masking*, where a tone at one frequency may make another tone at a different frequency inaudible or alter the perceived loudness of the second tone [16]. Audio input is divided into 32 frequency bands that are sampled independently. The range of frequencies in each band is not constant. Narrow bands are used for ranges where human perception is acute and wider bands are used where humans are less perceptive. The MPEG audio algorithm analyzes the relationships of the signals in the critical bands using a psychoacoustic model. Only those bands not masked by other bands are encoded in the bit stream. MPEG provides three compression layers, with increasing compression with increasing layers [83]. Independent of the compression layer, MPEG audio has four potential modes: (1) single channel, (2) two independent channels, (3) integrated stereo (two integrated channels into a single bit stream), and (4) integrated stereo exploiting redundancy between the two channels [16]. A transmission control scheme may choose to use any of the potential modes and compression layers. Depending upon the mode and compression layer used, MPEG may generate audio bit rates between 64 kbits/second and 256 kbits/second.

2.1.1.6. Other Techniques

There are many other audio coding techniques (*e.g.*, GSM [31], vocoders [11], professional quality music formats, *etc*.), but the techniques discussed represent the most common audio formats. Some vendors have introduced proprietary coding schemes. The proposed United States HDTV standard uses Dolby Labs' AC-3 digital audio compression, which provides six channels with an aggregate bit rate of 384 kbits/second to produce surround-sound audio quality [9]. The Intel DVI system [41] is of particular interest in this work since DVI is used in the conferencing systems described later. DVI supports PCM audio encodings [41]. Audio is captured, compressed, decompressed, and played directly by the DVI hardware and software. There is some flexibility in choosing the sampling rate of the audio, but generally little control of the audio capture and compression algorithms from outside the DVI system.

Audio samples are independent and can be played back without reference to preceding or following samples. The stereo bit rate is about 120-128 kbits/second.

2.1.2. Video Coding

2.1.2.1. Basic Compression Techniques

Digitized video produces a lot of data. For example, a common resolution for video conferencing is 256 × 240 picture elements (*pixels*). If each pixel is represented in RGB format with 8 bits per color component, an uncompressed frame is 256 × 240 × 24 = 1,474,560 bits. NTSC video has 30 frames per second, so the uncompressed bit rate for the conference is 30 frames/second × 1,474,560 bits/frame = 44,236,800 bits/second. Data rates of this size pose significant problems for existing computer systems. Many video compression techniques and algorithms have been developed to reduce the size of the video data stream. In this section, we survey a few of the more common compression techniques and some algorithms built from combinations of these techniques. Figure 2-4 provides a conceptual view of the video compression process. A *video compression algorithm* uses a combination of *video compression techniques* to produce a compressed bit stream from a digitized uncompressed video stream [16]. Compression techniques may be fixed or adaptive and may be grouped into the following categories: (1) simple, (2) interpolative, (3) predictive, (4) transforms, or (5) statistical.

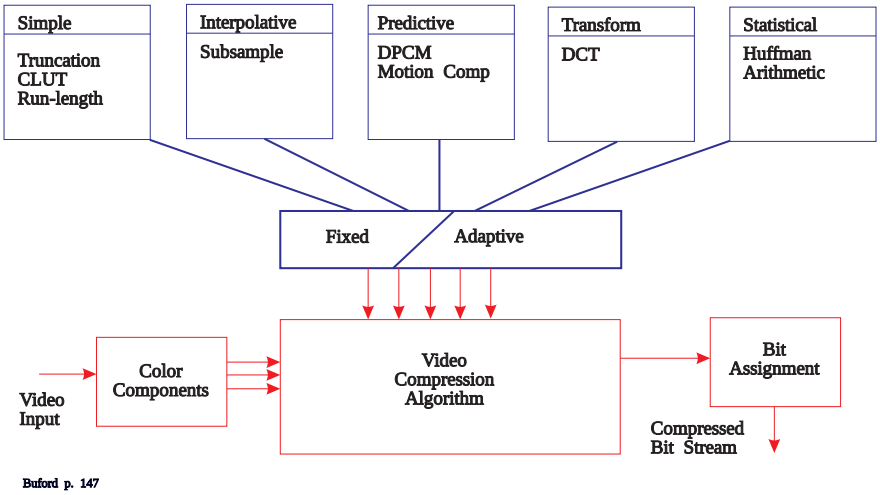


Figure 2-4: Video Compression [16]

Simple compression techniques attempt to reduce the video bit rate by direct manipulation of the bits. The *truncation* technique strips the least significant bits from the output stream, with a corresponding loss in image fidelity. The *color lookup table* (CLUT) technique maps each pixel to an index into a color table and transmits only the index. *Run length encoding* represents a string of repeated values by a single value and a length [16].

Interpolative techniques transmit a subset of the pixels in the image as complete samples and interpolates values for those pixels not transmitted from neighboring pixel values. Systems that represent images with separate luminance and chrominance components often use interpolative techniques. *Luminance* is the brightness of the pixels in an image and *chrominance* is the color associated with the pixels [16]. Because people are more sensitive to the luminance of an image than to the chrominance, the video signal is sometimes compressed by encoding the luminance and chrominance separately and giving the luminance component a larger share of the total signal than chrominance [16]. Often the full luminance component is recorded, but only a portion of the chrominance component. The chrominance values not recorded are estimated using interpolative techniques. When used in this way, the technique is sometimes called *color subsampling* [16]. Interpolative techniques are also used for generating successive approximations of the image, where a low resolution image is transmitted early and later refined by additional transmitted information [86].

We have already discussed *predictive* techniques such as DPCM and ADPCM. Predictive techniques are often used for motion compensation in the video stream. Elements from previous images are used to predict the values of later images based on a motion vector that maps a set of pixels in the old image to a set in the new. The changes between the old set and the new set are recorded as a set of differences.

Statistical coding or *entropy coding* techniques exploit some statistical feature of the data to reduce the number of bits required to represent the stream. Typically, this means assigning short bit strings to common patterns and longer bit strings to less common patterns. *Huffman encoding* [21] and *arithmetic encoding* [86] are the two most common statistical coding techniques used with video. Huffman codes use a table of pattern frequencies to map common patterns to short bit strings and rare patterns to longer bit strings. The decoder must have access to the same frequency table as the encoder. The frequency table may be pre-defined or built dynamically by analyzing the pattern frequencies in the input stream [21]. Arithmetic encoding is

similar to Huffman encoding, but arithmetic encoding uses an algorithmic statistical model to produce the codes rather than a frequency table. Arithmetic encoding is computationally more expensive than Huffman encoding, but often produces better compression and does not require coordination of tables between the coder and decoder [86].

Transformation techniques use some mathematical function to map the sample values into another set of transformed values. The transformed values are then compressed using one of the other techniques discussed in this section. The decoder restores the original samples by first decompressing the transformed values, then applying the inverse of the function to the decompressed values. There are many possible transformation functions, but the transformation only aids in compression if the function maps sample values into a set of transformed values that are easier to compress than the original samples. By far the most widely used transform in video processing is the *Discrete Cosine Transform* (DCT) [92]. We discuss the DCT in more detail in the next section.

2.1.2.2. Discrete Cosine Transform (DCT)

Most of the major video coding algorithms use the Discrete Cosine Transform (DCT). In competitive evaluation as part of the Joint Photographic Experts Group (JPEG) effort, DCT provided the best picture quality of all the compression techniques evaluated [113] and was subsequently adopted by JPEG (see section 2.1.2.3), H.261 (see section 2.1.2.4), and MPEG (see section 2.1.2.5). DCT is attractive not only as a result of the delivered picture quality, but also because fast algorithms exist for computing the DCT coefficients [92].

The DCT operates on an 8×8 group of pixel samples called a *block*. Each block is transformed using a 64 dimensional set of cosine basis vectors. The equations for the forward DCT (FDCT), $F(u,v)$, and inverse DCT (IDCT), $f(x,y)$, are given below [113]. In these equations, x and y are the row and column indices of the original block. The variables u and v represent the row and column indices for the resulting transformed block. Each transformed sample value, $F(u,v)$ where $u, v \in \{0..7\}$, is computed using the values in the original sample block.

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right]$$

$$f(x,y) = \frac{1}{4}\left[\sum_{x=0}^{7}\sum_{y=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right]$$

where: C(u), C(v) = $1/\sqrt{2}$ for u,v = 0; C(u), C(v) = 1 otherwise

The cosine basis vectors are orthogonal, so multiplying the basis vectors by some particular vector of constants and adding the results together can produce any vector in the 64 dimensional space [66]. The result of applying the FDCT is a set of 64 DCT coefficients, the constant multiples of the vectors, that encodes the original pixel values. One of the coefficients, the DC coefficient, is the coefficient associated with the constant basis function (u = 0) [86]. The remaining coefficients are associated with the remaining 63 higher frequency basis functions and are called the AC coefficients.

The block is decoded by applying the IDCT to the DCT coefficients. The decoder can calculate original sample values, $f(x, y)$, by performing the inverse mapping using the transformed sample values. In theory, the DCT encoding and decoding are lossless. In practice, the decoded block may have errors due to the computation of transcendental functions or representation errors [88].

The magnitude of the DCT coefficients may be larger than the magnitude of the incoming pixel sample. If all the DCT coefficients had to be explicitly encoded, it is possible the resulting number of bits would be greater than directly encoding the original pixel values. Fortunately, the DC coefficient alone provides a good estimate of the values in the original block of pixel samples. Higher frequency AC coefficients only incrementally improve the estimates of the original samples and AC coefficients usually have smaller magnitudes than the DC coefficient. Generally, the higher frequency AC coefficients have smaller magnitude than lower frequency AC coefficients. Higher frequency AC coefficients are often 0 or near zero. Compression algorithms take advantage of this fact and do not explicitly record each AC coefficient. Compression algorithms also apply other processes to the DCT coefficients to further compress the data. The following sections on specific coding algorithms discuss some of the more common techniques.

2.1.2.3. JPEG

The Joint Photographic Experts Group (JPEG) has developed a standard for still image compression [52]. Although not explicitly directed towards a continuous stream of images (*e.g.*, video conferencing), JPEG encoding can be used to produce a sequence of independently encoded video frames. JPEG has several operating modes: (1) sequential, (2) progressive, (3) hierarchical, and (4) lossless [52]. All modes except lossless mode are based on a DCT transformation. Lossless mode uses a form of DPCM to achieve perceptually lossless image compression, but only achieves approximately 2:1 compression [113]. This section focuses on the DCT-based modes because these modes achieve higher compression rates and are closely related to other video coding schemes (*e.g.*, H.261 and MPEG). The sequential mode is the simplest mode and is used here to describe a typical DCT-based encoding process.

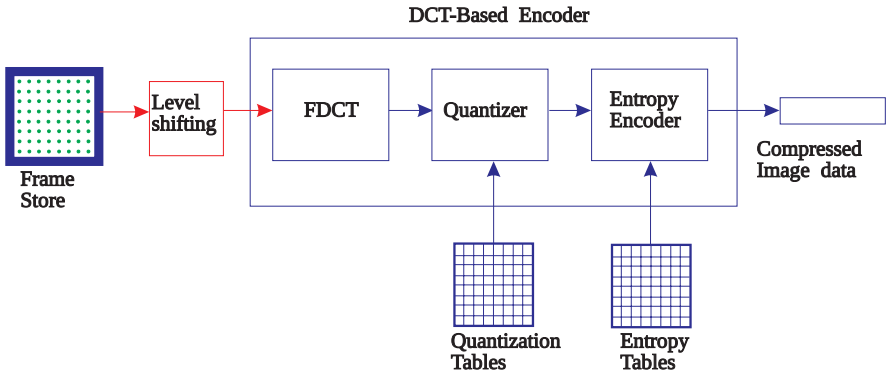


Figure 2-5: JPEG Sequential Mode Encoder (modified from [113])

The encoder processes each block in the image independently. Figure 2-5 illustrates the encoding process for a single block. The values of the pixel samples are shifted, if necessary, from a range of 0 to 255 to the range of –127 to 128 and are then passed to the FDCT. To this point, the process is nearly lossless (within the limits of the transcendental function and representation errors); however, to achieve greater compression, the encoder then quantizes the DCT coefficients.

In JPEG, *Quantization* is the process of reducing the magnitude of the DCT coefficients by dividing each coefficient by a specified step size. After division and rounding, many of the resulting quantized DCT coefficients have small or zero magnitudes and thus may be efficiently encoded; however, because of the division, the quantized DCT coefficients are accurate only within a step size, so the quantization

process is lossy and is the principal source of error with DCT-based encoders [113]. The *quantization table* holds the step sizes. Quantization tables are 8×8 tables and the step size in a particular cell of the quantization table is divided into the corresponding cell in the block of DCT coefficients. Ideally, the step sizes are chosen such that the loss of accuracy resulting from quantization has no perceptual impact. The encoder may use different quantization tables for different components of the image (*e.g.*, chrominance and luminance). The JPEG standard provides guidelines for quantization tables [86], but the application is free to substitute quantization tables of its choice. Furthermore, quantization tables may have an associated *q-factor* used to scale the entire quantization matrix. The encoder can change the *q*-factor to increase or decrease the compression ratio.

The encoder records the quantized coefficients from a block in a zigzag order (see Figure 2-6) [113]. DC coefficients between successive blocks are encoded using difference encoding (*e.g.*, DPCM). AC coefficients are run-length encoded. It is common for many of the high frequency AC coefficients to be zero after quantization, so AC coefficient encoding can be very efficient. The bit stream is then entropy encoded using either Huffman or arithmetic encoding.

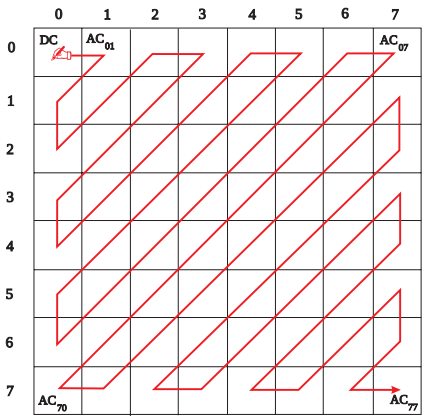


Figure 2-6: Zigzag Block Encoding Order (modified from [113])

Decoding (Figure 2-7) is the inverse process of encoding. The entropy decoder decodes the incoming bit stream. The decoder dequantizes the resulting coefficients and applies the IDCT. The result is the decoded block of pixels. JPEG specifies default entropy and quantization tables, but the tables may also be defined in the image

bit stream [52]. Since each frame is encoded independently, loss of a single frame does not affect display of subsequent frames.

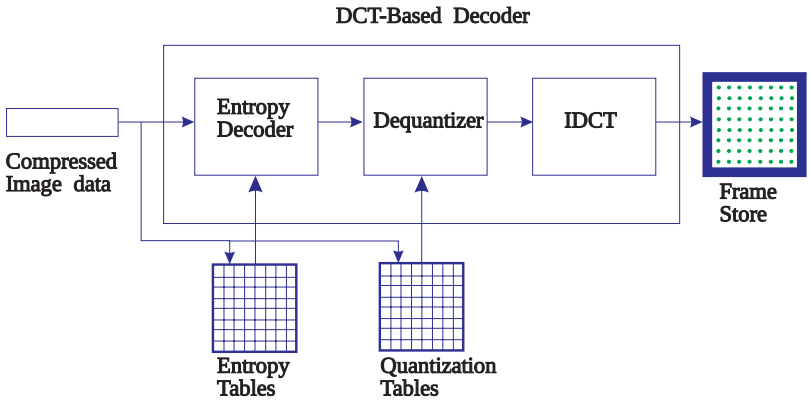


Figure 2-7: JPEG Sequential Mode Decoder (modified from [113])

Progressive mode is similar to sequential mode, but portions of the image are encoded in a sequence of scans rather than coding the entire image in a single scan. Progressive mode uses two complementary partitioning techniques. With *spectral selection*, the entire set of DCT coefficients are encoded as a series of subsets. A set of low frequency DCT coefficients are encoded first, then a set of higher frequency coefficients, and so on until the entire set of coefficients have been encoded. With *successive approximation*, the most significant bits of all quantized coefficients are encoded first and additional less significant bits are encoded later. Both of these techniques allow low resolution images to be displayed with low latency and refined over time with additional coefficients or less significant bits. A conferencing system could use either technique to reduce the video bit rate by transmitting a low resolution image (*i.e.*, the low frequency coefficient set or the most significant bits) and ignoring the successive scans.

Hierarchical mode encodes an image at multiple resolutions using a combination of predictive and interpolative techniques. Hierarchical encoding allows display of an image at multiple potential resolutions (*e.g.*, to support a variety of display devices). Similar hierarchical encodings have been used in video conferencing systems to accommodate multicast conferences where all conference destinations do not have the same available network capacity along the delivery paths [46].

2.1.2.4. H.261

H.261, also called *p*×64, is a CCITT standard for video conferencing over Integrated Services Digital Network (ISDN) channels [73]. The *p* in the *p*×64 may be from 1 to 30 and the H.261 standard covers codecs with bit rates in the range from 40 kilobytes/second up to 1.92 megabytes/second [23]. Unlike JPEG, H.261 is specifically designed for video conferencing. H.261 specifies the video encoding while related CCITT standards (*e.g.*, G.711 and G.728) cover audio and the multiplexing of audio and video (*e.g.*, H.221) [23]. H.261 can operate on standard ISDN services, where there are two 64 kbits/second data channels (B channels) and one 16 kbits/second control channel (D channel) [23]. Due to the focus on ISDN services, H.261 produces relatively low bit rate video when compared to other motion video encoding schemes such as MPEG.

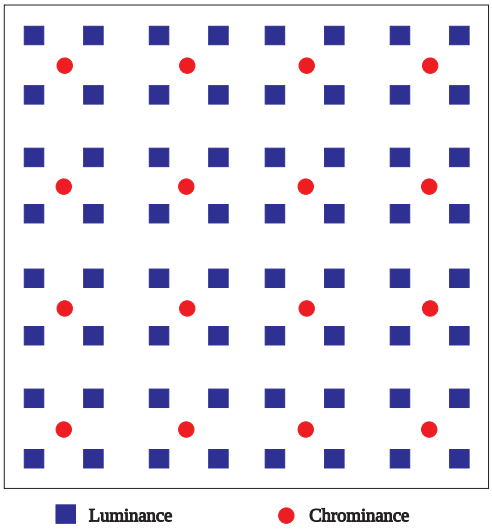


Figure 2-8: Resolution for Luminance and Chrominance in H.261 [73]

H.261 uses luminance-chrominance encoding and defines two picture formats, Common Intermediate Format (CIF) and Quarter-CIF (QCIF). All H.261 codecs must support QCIF; CIF is optional. QCIF has a 176 × 144 pixel image. CIF has a 352 × 288 image size. These sizes define the luminance component of the image. The chrominance component is encoded at lower resolutions and interpolated at the receiver to match the luminance resolution (Figure 2-8 shows the relative resolutions for one luminance block). The uncompressed bit rates for QCIF and CIF are 9.115

and 36.45 megabits/second, respectively. An H.261 codec must significantly compress the video stream to meet the limited ISDN bit rates.

H.261 uses both DCT and DPCM encodings. H.261 uses both *intraframe* encoding, where only a single frame is considered during encoding, and *interframe* encoding, where multiple frames are considered. Intraframe encoding exploits spatial redundancy in a frame. Interframe encoding exploits both spatial redundancy in a frame and temporal redundancies between frames [73]. Compression schemes could exploit temporal redundancies between frames at the pixel level by recording differences between corresponding pixel positions, but better compression rates are often achieved by performing motion compensation between frames. An H.261 encoder performs motion compensation on the basis of 16 × 16 *macroblocks* of pixels (4 luminance and 2 chrominance DCT blocks; see Figure 2-9).

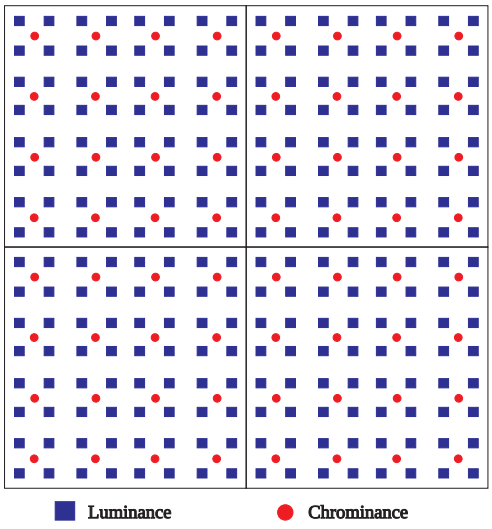


Figure 2-9: H.261 Macroblock [73]

The encoder logically tiles macroblocks onto an image and compares each macroblock with different 16 × 16 groups of pixels in the preceding image. The motion compensation algorithm only considers groups within a fixed radius of the position of the macroblock. The codec selects the best match based on comparison of the luminance component. The codec encodes the differences between the macroblock and the best match and records a motion vector that describes the movement of the macroblock. Although this motion compensation improves the compression of the video stream, it also increases the effects of loss on the stream. If the base frames are

lost in transmission, any subsequent difference or motion compensated frames cannot be played even if delivered.

Since H.261 is intended for video conferencing and thus must support real-time capture and display, the complexity of H.261 coders and decoders is similar. H.261 reqiures codecs to limit the combined encoding and decoding latency to less than 150 milliseconds. H.261 also attempts to provide automatic flow control based on the buffer occupancy on the sender. When the transmission buffer fills, the sender increases the quantizer step values to lower the generated bit rate. The goal is that the generated conference bit rate is roughly constant over time. H.261 can also adjust the conference bit rate by reducing the video frame rate. H.261 supports frame rates of 30, 15, 10, and 7.5 frames per second. Although intended for ISDN, schemes exist that allow packet-switched networks to carry H.261 [110].

2.1.2.5. MPEG

The Motion Pictures Experts Group (MPEG) standard is similar to H.261, but relaxes the low delay and low bit rate requirements. The primary requirement for the MPEG standard is the best possible quality at a given bit rate. There are four MPEG standards: MPEG-I MPEG-2, MPEG-3, and MPEG-4. MPEG-I provides roughly VHS quality video in a data stream of approximately 1.5 megabits per second. MPEG-2, MPEG-3, and MPEG-4 provide even higher quality at correspondingly higher bit rates. The MPEG standard includes both audio and video compression as well as synchronization between the audio and video streams. Due to the complexity associated with MPEG, a conferencing system may require hardware assistance to encode or decode the media steams in real-time.

MPEG was designed to support video-on-demand servers. As a result, MPEG has several features not present in H.261 and JPEG. MPEG supports random access points in a stored media stream, provides fast-forward and fast-backward support, supports reverse playback, and allows editing of the media stream. MPEG supports both asymmetric and symmetric encoding/decoding. Asymmetric encoding/decoding schemes are primarily for environments like video-on-demand, where the media stream is encoded only once, but decoded (played back) many times. In this environment, encoding need not occur in real-time and the compression algorithm may be optimized to provide increased compression, increased image quality, support for reverse

playback, *etc*. Symmetric encoding/decoding schemes are more appropriate for video conferencing since both encoding and decoding must both occur in real-time.

The basic MPEG algorithm is a DCT-based encoding with motion compensation. MPEG supports several frame types: (1) I - intra-coded frame, (2) P - predictive-coded frame, (3) B - bi-directionally predicted, interpolative-coded frame, and (4) D - fast search format. A stream may consist of different frame types and the sequence of frame types determines the type of motion compensation possible. We do not discuss D frames here since these frames are primarily for stored video services, such as video-on-demand. I frames are similar to a single JPEG image in that each I frame is independent of other frames. The receiver can decode play I frames without reference to other frames. I frames delimit a *Group Of Pictures* (GOP) in an MPEG video stream and provide the reference points for random access, fast searches, and editing. Each GOP is composed of an I frame and some number of following P and B frames (see Figure 2-10). P frames are predictively encoded based on a previous I or P frame. The prediction scheme is similar to the motion compensation in H.261. The codec compares macroblocks in the current frame with groups of pixels in previous frames. The best matches are then difference encoded and transmitted along with a motion vector. This results in P frames that are much smaller than I frames, reducing the overall stream bit rate, but at the expense of computation (*e.g.*, comparing groups of pixels with macroblocks and encoding the motion vector) and memory use (*e.g.*, the encoder must hold both the reference frame and the current frame in memory).

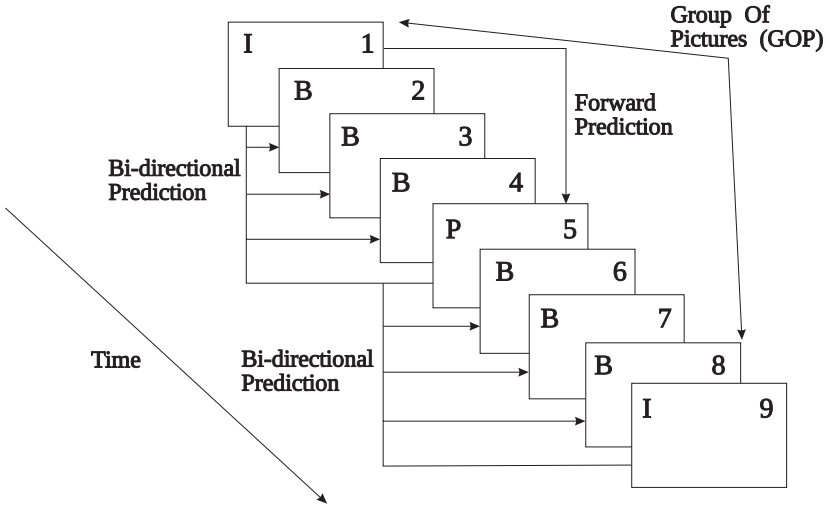


Figure 2-10: MPEG Frames [113]

B frames are similar to P frames, but are encoded based not only on preceding I or P frames, but also potentially on succeeding I or P frames. B frames may be (1) forward prediction encoded, where the B frame is based on a preceding I or P frame; (2) backward prediction encoded, where the B frame is based on a following I or P frame; or (3) interpolative encoded, where the B frame is difference encoded off an average of surrounding I and P frames. The compression for B frames may be very high since the motion compensation algorithm can exploit both past and future temporal redundancies using bi-directional encoding. From a video conferencing perspective, the primary disadvantage of B frame encoding is the latency associated with producing the B frame. The codec cannot encode a bi-directional B frame until the future reference frame is available. As a result, the codec may not encode the image associated with the B frame until several frame intervals after the image was captured. Also, since both the past and future reference frames must be present to decode a B frame, the transmission order of the frames is not the order of playback. For example, Figure 2-10 shows the logical order of a set of MPEG frames. In this example, each group of pictures is composed of a base I frame, 3 B frames, a P frame, and 3 more B frames. The logical order of the frames is IBBBPBBBI (see Figure 2-10), but the transmission order is IPBBBIBBB (see Figure 2-11). To decode the B frames, the destination machine must hold at least three frames in memory (*i.e.*, an I frame, a P frame, and one of the B frames) .

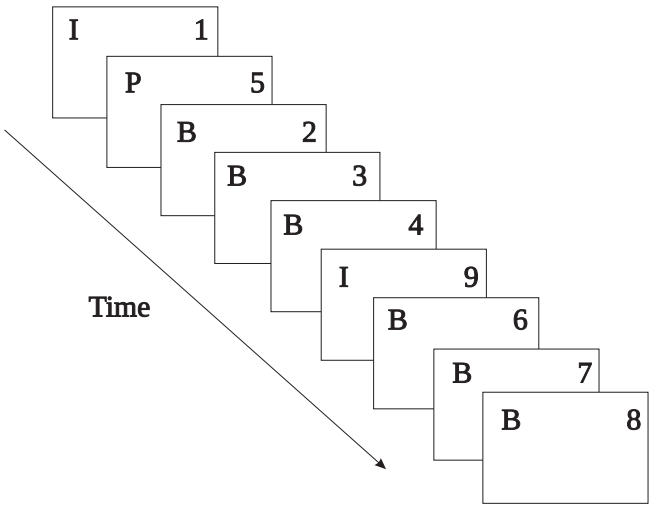


Figure 2-11: MPEG Frame Transmission Order

Packet loss may disrupt the playback of conference media streams and difference encoding schemes may exacerbate the effects of the packet loss. In the case of MPEG video, the receiver cannot decode P frames without the reference I frame. If the I frame is lost during transmission, the receiver cannot play the associated P frames even if the P frames are delivered intact. Similarly, delivered B frames, depending upon the encoding scheme used, may be worthless if logically preceding or succeeding I or P frames are lost in transit. In the case of video conferencing, use of B and P frames reduces the media bit rates at the risk of potentially increasing the effects of packet loss on conference quality. MPEG allows the application to control the use of I, B, and P frames, so the application must choose an appropriate frame compression strategy for the particular task and environment.

MPEG-2 defines fixed-length packet sizes (4 byte header and 184 byte payload) for video, audio, and auxiliary data. Each packet contains only one type of data (e.g., audio, video, or auxiliary). An MPEG-2 aggregate stream consists of an arbitrary mixture of media streams. There is no predefined mix of packet types for an aggregate stream and the composition can adjust dynamically. The packet header contains a 13 bit packet identifier, a field describing the type of data present in the packet, and presentation time stamps for lip synchronization [9]. The flexible format of a MPEG-2 program makes MPEG-2 an attractive format for scaling. Video can be scaled at the sub-channel level as well as in the frequency or temporal dimensions available in MPEG-I. MPEG-2 is the basis for the video portion of the proposed United States and European HDTV standards [9, 36], which implies significant potential for interoperability between computers and televisions [9].

2.1.2.6. Digital Video Interactive (DVI)

As mentioned earlier, the Digital Video Interactive (DVI) encoding from Intel and IBM is of particular interest in this work because the conferencing systems used for the experiments in the later chapters use DVI technology. DVI is a combined hardware and software solution [44, 93]. The Intel/IBM ActionMedia I and ActionMedia II controllers provide most of the media stream processing directly on the controller using a set of proprietary algorithms. The original version of the DVI hardware and software had limited flexibility and was intended to function much like a digital VCR [44]. Later versions of DVI are based on a “digital production studio” model and are considerably more flexible than the original DVI implementation [41].

DVI supports two proprietary video formats: Production Level Video (PLV) and Real-Time Video (RTV) [41]. PLV is intended for off-line compression of video and real-time decompression. PLV performs interframe encoding including motion compensation and provides high-quality video output [44]. PLV uses a form of video encoding called *region coding* where regions of the image are decomposed into image primitives. The system provides the implementation of the image primitives and efficiently decodes and displays the primitives. Although playback is efficient, region identification and motion compensation make compression slow and expensive [74]. In contrast, the system can capture and compress RTV in real-time (at the cost of picture quality when compared with PLV), so RTV is more suitable for video conferencing applications. RTV uses intraframe encoding exclusively [16] and is similar to JPEG, although the specific algorithms are proprietary. Intel has subsequently sold the DVI technology to Horizons Tech [82] and has focused on a software-only derivation of DVI called Indeo. Indeo is a vector quantization-based algorithm (see section 2.1.2.6) with intraframe compression and interframe motion compensation.

2.1.2.7. Other Algorithms

There are many other algorithms for compression video. Some video conferencing systems use *vector quantization* (VQ) to represent video. Vector quantization divides the picture into blocks and uses the coefficients describing each block as a vector. The codec divides the entire vector space into regions and calculates a representative vector describing all the vectors in a particular region. The coder records the representative vectors in a code book and passed the code book to the decoder. Coding consists of mapping blocks to the closest representative vector and assigning the associated vector label to the block. Decoding consists merely of looking up the vector label in the code book and displaying the representative vector [112]. Although decompression of a vector quantized image is simple and fast, vector quantization compression is computationally expensive [114]. The asymmetric nature of VQ makes the technique more appropriate for storage and playback of video than for video conferencing.

Another compression technique is subband coding [112]. Subband codecs split the video signal into spectral components and code each spectral region of the image as a separate band. This type of encoding is similar to the spectral selection coding in JPEG, although the actual coding of a band may not be via the DCT (*e.g.*, a subband

may be encoded using vector quantization) [74]. The resulting stream is naturally hierarchical and thus well suited for the hierarchical scaling techniques discussed below. Like VQ, subband encoding can be computationally expensive and may require hardware assistance to perform adequately, particularly in real-time applications [112].

The compact disc-interactive (CD-I) standard uses video encodings similar to MPEG, but is primarily concerned with playback of stored video rather than video conferencing [99]. The JBIG standard provides for lossless encoding of greyscale images; however, since lossy video compression techniques typically produce significantly better compression, JBIG is not used for video conferencing [5]. In the personal computer world, Microsoft has a video standard for the Microsoft Windows environment called Video for Windows [87]. Similarly, Apple has a multimedia standard called Quicktime for the Macintosh environment that provides a software architecture and a set of tools for manipulating still images, audio, and video. An Image Compression Manager (ICM) shields applications from the specifics of the compression techniques [16, 89] .

2.2. Scaling Techniques

2.2.1. Introduction to Media Scaling

One of the ways a video conferencing application can adapt to the effects of network congestion is to lower the bit rates of the media streams when congestion is high and later increase the bit rates when congestion subsides. The conferencing application can thus *scale* the bit rates of the media streams by adjusting the parameters associated with the generation and compression stages. Scaling may either increase or decrease the media stream bit rate, depending upon the circumstances, and is used to adapt the media stream bit rate to either (1) the capabilities of a destination node or (2) the current network environment. In this work, we are interested in using scaling as a tool for adapting to network conditions. Our focus is on *transmit scalability*, where the coder does the scaling, rather than *receive scalability*, where the decoder does the scaling [74].

Scaling and compression techniques are closely related. Compression is concerned with reducing the size of media data units. Scaling is concerned with dynamically changing the number and size of the transmitted media units based on the state of the network. Scaling affects which media data units are transmitted, but not necessarily

which media units are captured. Scaling may involve changing compression schemes or compression parameters or may be independent of the compression scheme (*e.g.*, transmitting only one channel of audio from a multi-channel audio source). Since the bit rate associated with a video stream is typically much larger than that associated with an audio stream, scaling is more commonly applied to the video stream than to the audio stream. In addition, humans are much more sensitive to scaling in the audio stream than to the video stream.

Delgrossi, *et al.*, categorize scaling techniques into *transparent scaling* and *non-transparent scaling*. Transparent scaling techniques allow the transport layer to scale the media stream without the knowledge or involvement of the application producing the media streams. Non-transparent scaling techniques require interaction between the transport layer and the application to adapt the media stream [27]. Transparent scaling schemes often involve hierarchically encoded streams. The media stream may be a single, multi-layer stream or perhaps multiple complementary streams. Multi-layer streams are composed of several encodings of the media data. Each encoding is at a different resolution. Higher resolution encodings may be formed by enhancing lower resolution encodings or may be entirely independent of the low resolution encodings. The transport mechanism is aware of the structure of the stream and reduces the bit rate of the stream by transmitting only a portion of the entire media stream (*i.e.*, a subset of the lower resolution encodings) when the network is congested . When multiple complementary streams carry the media stream, each of the component streams encode the media at a different resolution. The transport mechanism scales the entire media stream by only transmitting a subset of the component streams (again, the low resolution encodings) when the network is congested.

With non-transparent scaling, the transport mechanism directly or indirectly modifies some parameters (*e.g.*, coding scheme or frame rate) associated with generation of the media stream. The advantage of non-transparent scaling over transparent scaling is that the transport layer need not understand the actual format of the media stream. The disadvantage of non-transparent scaling is that the transport layer and media-generating application must cooperate to adapt the media bit rate. In transparent scaling the transport layer can act in isolation.

2.2.2. Classification of Scaling Techniques

This section discusses a few of the most common scaling techniques used with conferencing systems. *Temporal scaling* [27] reduces the bit rate of the media stream by selectively removing media frames from the stream. This technique does not work well with audio since even small audio gaps are noticeable by humans, but may be effective with video, particularly when video frames are coded using some frame-independent coding scheme (*e.g.*, JPEG images or MPEG with only I-frames). Temporal scaling is easy to implement since the conferencing application can simply periodically drop generated frames. For example, a system using an MPEG codec can implement temporal scaling by selectively dropping generated B frames from the transmitted stream.

Spatial scaling [27] reduces the number of pixels in a video image. Spatial scaling may be simply transmitting a smaller image with fewer pixels, but more often the codec provides spatial scaling by selectively subsampling the image [16, 18]. The image is subsampled and the missing pixels decoded by interpolating between the transmitted pixels. The size of the image remains constant. The codec may provide parameters that affect the level of subsampling and thus allow dynamic scaling of the image via changes in the parameter values. Naturally, the degree of subsampling directly affects the quality of the played image.

Frequency scaling [27] reduces a video bit stream encoded by a DCT transformation by reducing the DCT coefficients associated with the blocks composing the image. With the DCT, the content of a block is primarily encoded in the DC coefficient. AC coefficients refine the values in the block. Lower frequency AC coefficients usually have more effect on the final decoded value of the block than do higher frequency AC coefficients. High frequency coefficients can often be removed from the media stream with little impact on the perceived quality of the decoded image. Frequency scaling exploits this fact by not transmitting the high frequency coefficients. For example, JPEG spectral selection encodes low frequency DCT coefficients before those with high frequency. Using a JPEG codec, a conferencing application could perform frequency scaling by transmitting only the low frequency coefficients and ignoring the high frequencies.

Amplitudinal scaling [27] also operates on the DCT coefficients and scales the stream by reducing the magnitude of the coefficients. This is usually done by increasing the

quantization step size, either by modifying the quantization tables or adjusting the *q*-factor. For example, H.261 uses amplitudinal scaling via quantization as its primary tool for controlling the bit rate of the video stream [73, 112]. JPEG's successive approximation is another potential technique for amplitudinal scaling.

Color space scaling [27, 112] adjusts the bit rate by either reducing the number of bits representing color for each pixel in the image or reducing the chrominance component of the image. This technique is particularly appropriate for luminance-chrominance color representations since humans are much more sensitive to the luminance component than to the chrominance [112]. For example, the conferencing system may transmit four luminance blocks with only two chrominance blocks. When decoding, the values in the chrominance blocks are interpolated across all four luminance blocks (see the H.261 macroblock example in Figure 2-9).

Hierarchical scaling transmits only a portion of a multi-part media encoding. For video, the multi-part encoding is typically a hierarchical image consisting of several versions of the source image subsampled at different levels. The hierarchical image may consist of several independent images at different resolutions or a set of progressive images. Progressive images add successive levels of detail to an original low resolution image to obtain higher quality images. For audio, the multi-part encoding is usually composed of multiple audio channels. A conference source may scale audio by transmitting only a subset of all the audio channels (*e.g.*, one channel of a stereo stream). Hierarchical scaling may be either transparent or non-transparent and is one of the few scaling techniques that is perceptually viable for audio. What we call hierarchical scaling here is considered part of spatial scaling in [27], but we prefer distinguishing between the two since hierarchical encodings need not be formed from spatial subsamples. For example, audio hierarchical encodings may be based on channels and video hierarchical encodings may be based on DCT frequencies. Some coding schemes are inherently hierarchical. For example, MPEG-II defines a hierarchical coding scheme with three layers of image quality [27]. Even if a basic algorithm is not inherently hierarchical, it may be possible to impose a hierarchy on the scheme. For example, two-layer hierarchical encoding schemes have been proposed for H.261 [112]. Hierarchical encodings may be carried by one or more transmission streams. When a hierarchical stream is carried by a single transmission stream, transparent scaling requires the transport layer to understand the format of the hierarchical encoding. When different layers of the hierarchical encoding are carried as

independent streams, the set of all component streams is sometimes called a *scalable flow* [46]. Hierarchical scaling on a scalable flow consists of transmitting only a subset of the streams in the flow. Implementing transparent scaling is easier with scalable flows since the transport layer deals with individual streams rather than the hierarchical encoding structure [27, 46].

Motion scaling changes the bit rate of a motion compensated stream by adapting the thresholds and search neighborhoods used for motion detection [112]. Generally, lower motion detection thresholds and larger search neighborhoods produce lower transmitted bit rates since the codec can exploit more temporal redundancy between images. On the other hand, the codec may spend more processing time (and elapsed time) compressing the image.

2.2.3. Perceptual Effects of Scaling

Delgrossi, *et al.*, considered transparent and non-transparent scaling from the perspective of the conferencing application and its relationship to the transport layer. There is a second transparency issue with scaling which is the effect of scaling on the perceived quality of the resulting media stream by a human. We say a scaling technique is *perceptually transparent* if humans cannot perceive the change in media quality before and after the scaling change. Unfortunately, given the nature of scaling, it is often difficult to achieve perceptually transparent scaling. Scaling often leads to a reduction in the transmitted bit rate of a media stream and this reduction usually has a negative impact on the quality of the delivered media stream. Alternately increasing and decreasing the media bit rate may also affect the perception of the conference quality. Unfortunately, although some have suggested measures for video quality [107] and perceptual studies have been done on video coding quality [113], we are aware of no work that compares the effect of dynamic changes in media fidelity on perceived conference quality. It is likely that changes in media fidelity have a negative effect on conference quality, but it is important to remember that media scaling is performed only in reaction to a restriction in the transmission or playout of the media stream. The effect of media scaling on human perception must be weighed against the effects of not performing scaling. In many environments, not performing scaling leads to network conditions that severely limit media delivery and where the delivered media quality is poor. Media scaling may have an adverse impact on perceived conference quality, but the absence of scaling may result in significantly worse perceived quality.

2.3. Network Support for Live Continuous Media

This section surveys some of the techniques and systems developed to support transmission of live continuous media across computer networks, particularly for video conferencing applications.

2.3.1. Dedicated Channel Systems

A major problem associated with transmitting a video conference across a network is the degradation of conference quality because of competition for resources within the network. A way to eliminate this problem is to eliminate the competition for resources by dedicating the network resources to the conference. A dedicated physical communications path, such as an ISDN or leased telephone line, can be used to carry the conference data streams. We call such systems *dedicated channel systems* and several video conferencing systems use this strategy, most notably those from the telephone companies and those from companies specializing in room-based or standalone video conferencing.

Videophones have been available since the 1960s [78] and are available from several telephone companies, such as AT&T [5, 23] and MCI [16]. Videophones offer high quality audio and adequate video; however, despite intense marketing, videophones have remained a small market [78]. The lack of acceptance of the videophone is partially caused by infrastructure. Both ends of the connection must have compatible videophones and videophones are not widespread. However, it is more likely that videophones failed in the marketplace because adding video to a conversation, particularly a routine person-to-person conversation, simply does not provide enough added value over an audio-only conversation. Computer-based conferencing, on the other hand, is a richer communication environment since it potentially adds all the tools of the computer, in addition to audio and video, to the conversation [58]. This is the primary reason that computer-based video conferencing may prove more commercially successful than the videophone projects.

Room-based or standalone conferencing systems have been more successful than videophones because corporations are able to share video conferencing facilities among a large population, thus spreading the infrastructure costs. Also, standalone conferencing systems are typically used for group meetings and presentations rather than routine person-to-person conversations, so the value of the delivered video is

enhanced. All the commercially available standalone conferencing systems rely on dedicated network links to deliver consistent conference quality.

There are many providers of standalone conferencing systems, but the market is dominated by Compression Labs, PictureTel, and VideoTelecom in the United States and British Telecom in Europe [63]. Compression Labs uses one of three compression algorithms, one based on H.261 and two, CTX and CTX Plus, based on a proprietary algorithm called Cosine Transform Extended. Depending upon the algorithm, Compression Labs supports video frame rates of 15 to 30 frames per second with data rates between 56 kbits/second to 2 megabits/second. CTX Plus offers the highest frame and bit rates. Video Telecom supports up to 15 frames per second, with bit rates in the range of 56-786 kbits/second, using H.261. Video Telecom provides up to 30 frames/second using a proprietary algorithm called Blue Chip [63]. PictureTel also supports H.261, but achieves better performance with two proprietary vector quantization-based algorithms called SG2 and SG3/HQV. PictureTel offers frame rates up to 10 frames per second with bit rates in the 56-768 kbits/second, but at lower cost than Compression Labs and Video Telecom [63, 112]. British Telecom uses H.261 exclusively and supports frame rates up to 30 frames/second with bit rates between 56 kbits/second and 2 megabits/second [63]. Most of the video conferencing systems compress audio using ADPCM algorithms and PictureTel has the reputation of having particularly good algorithms for echo cancellation [63].

Standalone conferencing systems provide high quality video conferences. Unfortunately, like videophones, standalone systems are not usually integrated with corporate or personal computer systems. Standalone systems are also usually shared among a relatively large population, which limits the usefulness and availability of such systems for *ad hoc* or continuous collaboration. In addition, the requirement for dedicated network facilities makes dedicated channel systems expensive, particularly considering the fact that most collaborators are already connected via some existing computer network.

2.3.2. Reservation-based Control

Dedicating network resources eliminates the interference of competing traffic from a video conference, but the expense of dedicated channel systems and the lack of integration with existing computer networks has led to much interest in techniques for providing the illusion of dedicated resources across a shared computer network.

The basic idea behind all these *reservation-based control* techniques is that a video conference (or more generally, any real-time application requiring performance guarantees) makes a reservation request at connection establishment time for the entire logical path between source and sink. The *reservation request* consists of a *flow specification* [118] that describes the characteristics of the data stream (*e.g.*, peak and average bit rates) and a *performance specification* that defines the level of performance required for the session (*e.g.*, maximum delay). Performance specifications are usually based on throughput, delay, jitter, and packet loss [32]. The reservation request is used to determine if each network element along the logical path can accommodate the stream requirements. If so, each network element reserves sufficient network resources (*e.g.*, link capacity, router buffers, *etc.*) to guarantee that the performance along the path meets or exceeds the requirements specified in the reservation request. If any element in the path cannot provide the requested level of service, the reservation request is rejected. The protocol to reserve resources along the path is called the *reservation protocol.* The process of determining whether a new resource reservation can be accommodated is called *admission control*. After a connection is admitted, each network element must follow a *service discipline* [27] to achieve its committed level of performance. Typical components of a service discipline include resource allocation and scheduling algorithms for packet queues, buffers, CPU, link capacity, link access priority, *etc.* Controlling any single resource in isolation is generally insufficient to guarantee performance levels, particularly on multi-hop network paths. Some of the proposed reservation-based schemes include routing and transport protocols, while others rely on existing protocols. Theoretically, the underlying routing protocols could assist in providing guaranteed performance levels, but exploiting this potential remains largely a research topic [12, 118].

ST-II [108] is one of the earliest of the proposed reservation-based schemes. ST-II is a full internetwork protocol (*i.e.*, like the Internet Protocol (IP) [20]) and handles both data delivery and network control [27]. ST-II streams are simplex (*i.e.*, data travels in only one direction on the path) and connection-oriented. All stream packets follow the same path from the source to the sink. ST-II defines a reservation request format and a reservation protocol. The stream sender initiates reservations at connection setup. Each network element in the path receives the reservation request, reserves resources as necessary, updates the request with information about the performance guaranteed at the element (*e.g.*, maximum delay), and forwards the updated request to the next element in the path. When the receiver receives the request, it determines if the

resources reserved along the path are adequate for the conference by comparing the desired performance guarantees to those actually guaranteed. If the specification is satisfied, the connection is admitted; otherwise, it is rejected. The receiver sends a message with the acceptance status of the session back to the sender. Multipoint connections are built as a tree of connections with the sender as the root of the tree. Once established, network elements are aware of the connection and this awareness lasts for the life of the connection. In this case, we say the network elements have "hard" state that need not be refreshed [27]. Since ST-II has protocols to route and transport the data, connection identifiers can be used in routing decisions. Implementations of ST-II exist [84] and reservation-based systems, such as the Heidelburg Transport System (HeiTS) [27], have been built on top of ST-II; however, ST-II does not specify the required resource disciplines for the network elements. Service disciplines such as Golestani's *Stop-and-Go Queueing* [40, 109], that provide deterministic guaranteed levels of performance, could be integrated within the ST-II framework, but selection and implementation of a service discipline is left to the ST-II implementer [8].

The ReSerVation Protocol (RSVP) [118] provides some of the same functions as ST-II; however, where ST-II is a complete internetwork protocol, RSVP is solely a reservation protocol, relying on some other protocol, such as IP, for data transport. Unlike ST-II, RSVP does not define a flow specification, but instead treats the flow specification as an uninterpreted byte stream. The reservation protocol is receiver-initiated in RSVP and senders are not aware of the stations receiving their transmissions. The rationale for choosing receiver-initiated reservations versus sender-initiated reservations is that the receiver is better aware of its own capabilities and its own quality needs than the sender [118]. Receiver-initiated reservations also relieves the sender of some of the overhead involved in multicast transmission. This improves scalability since senders may be able to support a larger number of receivers. RSVP assumes the receivers have some means of identifying a path to a particular sender. Intermediate nodes store connection reservation information as "soft" state that must be periodically refreshed by the receiver [27]. RSVP supports both point-to-point and multipoint connections. Since RSVP does not actually carry the data, RSVP reservations must be somehow mapped to the underlying data transport services, such as IP connectionless services [27]. Furthermore, RSVP does not attempt to define a service discipline for implementing the reservations communicated via RSVP.

An interesting feature of RSVP is the concept of a receiver-specified *filter*. Filters may be sent as part of the specification and identify the portions of the sending stream to propagate downstream toward the sender [118]. This allows, for example, receiver-specified hierarchical video scaling within an intermediate node. The receiver, which presumably better knows its display capabilities than the sender, installs a filter at some intermediate node that extracts a low resolution image from the full sender stream and forwards only the low resolution image to the receiver. A receiver may use a filter to specify reservations on a capacity basis rather than content, so receivers can "switch channels" without changing the filter. RSVP supports reservations with no filters, fixed filters, and dynamic filters [118]. Multiple receivers can share resource reservations for the same data stream and the multiple filters can be combined as aggregate reservations at branch points in the transmission tree.

Perhaps the most complete implementation of a reservation-based system is the Tenet suite of protocols [33]. The Tenet suite defines a set of performance measures for stream reservations [32], a reservation protocol and admission control algorithm [33], a set of protocols for real-time data transport that coexist with IP [8], and a set of service disciplines to realize the performance guarantees. The sender initiates reservations. The Tenet protocols reserve resources for the best possible service on the forward pass and relax the constraints, if possible, on the return path [33][1]. The current Tenet protocols provide deterministic performance bounds if the client (sender) abides by the negotiated reservation contract. The Tenet group plans to support statistical or fractional bounds in future implementations. Current schemes limit streams to a single fixed routing path, but the group plans to support dynamic routing in the future. The Tenet suite has been implemented in the Sequoia 2000 network using T1 and FDDI network links and was shown to be superior to naive UDP transmission in delivering audio and video streams over congested networks [7].

A number of rate-based control schemes have been proposed with potential application to audio and video conferencing. Rate control schemes are particularly popular for large ATM networks where the high network bandwidth $\times$ delay product makes traditional flow control schemes less effective [29, 40]. Rate-based schemes usually seek to manage the data streams introduced at a source so that the streams

[1] ST-II does not relax reservations made on the forward pass [27]

obey some negotiated peak and average rates. Rate-based schemes such as the "leaky bucket" [12] have been proposed to smooth bursty continuous media traffic to match some constant bit rate (CBR) channel. Rate-based schemes are not in widespread use because of the limited deployment of ATM networks.

Reservation-based control techniques offer the performance guarantees of dedicated channel systems, but without the expense of dedicated hardware. Unfortunately, reservation-based schemes can still be expensive. Providing deterministic guarantees for bursty streams can lead to poor utilization of the network since reservations along the path must be made for peak rate requirements [67]. Statistical or fractional guarantees [67] offer the potential for improved utilization of the network by exploiting the statistical nature of bursty streams in aggregate, but providing statistical or fractional guarantees has proven a difficult challenge in real-time processor scheduling and is likely to prove equally challenging for resource allocation problems along a network path. Furthermore, providing network guarantees requires that a distributed algorithm, the admission control protocol, be executed along the network path to determine if a connection can be admitted. This implies computation within network elements along the path. This may be feasible with new networks built with new network technologies such as ATM. Unfortunately, many current networks are built from network elements that do not support reservation algorithms. Furthermore, most existing network elements (*e.g.,* routers, bridges, gateways) do not support service disciplines providing guaranteed performance. Moreover, service disciplines that maintain predictable traffic flow and limit delay jitter across the entire path use relatively sophisticated algorithms [67, 40]. The result of all these challenges is that few networks currently support deterministic guarantees and still fewer support statistical or fractional guarantees [7, 8].

2.3.3. Best-Effort Systems

According to Van Jacobson [55], there are two schools of thought on resource reservation for multimedia communications. One is that resource reservations are absolutely necessary. Dominico Ferrari is perhaps the most eloquent spokesman for this position:

> "In our opinion, most of the clients concerned about real-time performance require mathematically provable, quantitative, though not necessarily rigid or deterministic guarantees...We believe the only way to provide predictable

performance is to offer *a priori* guarantees to the client, and then ensure that these guarantees are not violated." [33]

The second school of thought holds that reservation is unnecessary. According to this position, adaptive techniques, with layered encodings to supply a range of bit rates, can sufficiently ameliorate the effects of resource competition such that the results are not noticeable or are within tolerable ranges. Prototype systems are being built based on both approaches, but according to Jacobson there is no conclusive answer yet and the eventual solution is likely to be combination of both sets of techniques [55].

We call transmission control schemes that do not reserve resources, and where conference quality varies based on the current actual demand for the network resources, *best-effort* systems. We agree with Jacobson's claim that both reservation-based and best-effort techniques will be employed, either separately or in concert, in the future. We contend that, for the particular case of audio and video conferencing, as long as it costs more to provide guaranteed levels of service than best-effort services, best-effort systems will persist. The cost may be in subscription costs, per call costs, equipment costs, software complexity, or a variety of other measures. Of course, there are applications that will require guaranteed performance (*e.g.*, medical consultation systems), but as long as there is a cheaper best-effort service, a large segment of the video conferencing users will opt for that service. Even Ferrari has found it practical to produce a best-effort system to support Ethernets, since no performance guarantees can be made across a CSMA/CD system [18].

There are many research and commercial best-effort video conferencing systems, but we will discuss only a few of the better known systems. The CU-SeeMe video conferencing system was developed at Cornell University. CU-SeeMe directly supports point-to-point connections and supports multicast connections using a *reflector*. The reflector is a program running on a node accessible to all parties in the multicast. Rather than directly connecting to each participant in a multicast conference, each user connects to the reflector. The reflector accepts video from each participant and distributes the video to all conference participants. CU-SeeMe does not adapt to network conditions. The user may select a desired outbound bit rate for the video and video is scaled to approximately match this rate through an unspecified scaling technique. CU-SeeMe delivers reasonable quality video when the network is lightly loaded. Unfortunately, when network congestion is high, CU-SeeMe does not

perform well, with high delivered stream latencies (measured in seconds) and low delivered frame rates [117].

The nv [37] system from Xerox PARC is another research system that provides gray scale images. The system uses a proprietary compression algorithm [31] and delivers a maximum of 10 frames per second (with an average of 3-5 frames/second being typical) and a default bit rate of about 128 Kbits/second [78].

The IBM Person-to-Person system is a commercial offering that provides a full collaborative environment, including audio and video conferencing, shared electronic whiteboards, file transfer facilities, *etc.* The audio and video system is based on the ActionMedia II hardware and software (DVI). The system supports both dedicated channel (via ISDN) and best-effort modes (via LANs). Person-to-Person supports several transport protocols (*e.g.*, APPN, UDP, NetBIOS, IPX). Video is full-color and capable of high frame rates and low latency in unloaded networks. Unfortunately, the network load directly affects the delivered quality. Furthermore, the relatively high bit rates associated with the video stream can have a severe impact on LAN performance and has led some groups to limit use of the video stream [117].

Like Person-to-Person, Intel's ProShare conferencing product [50] is a full collaborative environment and supports both dedicated channel (via ISDN) and best-effort modes (via LANs). ProShare uses Indeo video compression. GSM, an audio compression algorithm used for cellular phone systems in Europe, is used for audio compression. The combination of these two algorithms gives a simplex combined audio/video bit rate of about 100 Kbits/second. ProShare supports the transport protocols UDP, IPX and NetBIOS. In both the ISDN and LAN modes, the receiver buffers frames to ameliorate the effects of network jitter. With ISDN, ProShare uses the D channel only for call setup. ProShare dynamically allocates the two B channels (64 kbits/sec each) between video and data with 16 Kbits/second reserved for audio. Data has priority over video and video degrades with increased data traffic, while audio is unaffected. Video can use a maximum of 90 Kbits/second. On LANs, a unique aspect of the ProShare system is the LANDesk Personal Conferencing Manager. This software runs as a server on a LAN and limits the LAN resources consumed by ProShare conferences to some customer-specified level. All conferences contact the Personal Conferencing Manager before establishment of a connection to determine whether the conference is allowed given the current load and number of active conferences on the LAN. If ProShare cannot transmit all of the

offered data because of competition on the sender's network, ProShare favors audio packets over video for transmission and video frames may be delayed or dropped at the sender. Unfortunately, this technique only helps if the network congestion is on the link to which the sender is directly connected.

In addition to a reservation-based implementation, the Heidelburg Transport System (HeiTS) [27] also supports a best-effort service using media scaling based on feedback from the receiver. The HeiTS best-effort system uses a rate-controlled transport system based on media frames. Frames have expected arrival times and late arrivals signal that a bottleneck is present. Expected arrival times are based on previous arrivals and the frame generation period. The first reaction to congestion is to throw away late or excess packets. If the number of late packets exceeds a specified threshold, feedback signals the sender to throttle the bit rate of the offered traffic. Packets have "importance" tags and the network discards packets based on these tags (*i.e.*, least valuable first). If congestion continues to get worse, eventually the connection is terminated. The decision to scale up is based on heuristics; the stream is scaled up after an interval with stable performance. For multicast conferences, the worst delivery stream may determine the offered stream for all. When operating in concert with the reservation-based HeiTS system, the best-effort system may use what HeiTS calls *discrete scaling* [27] where a low resolution stream is guaranteed. This ensures a minimum quality level. The low resolution stream is augmented with sub-streams sent without guarantees. Since sub-streams are transmitted independently, different parts of the multicast tree can potentially scale differently at the sub-stream level.

Sun Microsystems has implemented a best-effort scheme that uses hierarchical encodings to adapt to network congestion [46]. Like RSVP, the Sun system uses filters at intermediate nodes to determine which sub-streams are forwarded to a particular destination. Filters may be dynamically or manually set (*e.g.*, by a network administrator) at known bottlenecks. The filters may use frequency (*e.g.*, spectral encoding or successive approximation), spatial (*e.g.*, subsampling), or temporal (*e.g.*, dropping B and P frames) scaling. Intermediate nodes are aware of the stream structure and may discard portions of the streams when congestion is present. Since intermediate nodes implement the scaling, local conditions and the relative importance of the streams are enough to determine the required scaling and so there is no need for feedback. Most transmission control is done within the network, but there may need

to be admission controls to govern incoming streams. Of course, intermediate nodes must support filters for this scheme to work.

The INRIA Video System (IVS) [14, 49] is based on H.261 video. H.261 was intended for ISDN, but IVS has defined a packetization scheme to allow H.261 to be used over packet-switched networks [110]. IVS uses the Real-time Transport Protocol (RTP) [97] transport mechanism. RTP supports end-to-end stream delivery with real-time characteristics and works on top of IP, ST-II, or UDP. IVS supports multi-party or person-to-person conferences. IVS attempts to operate at a target bit rate and video is scaled to achieve the target rate. IVS scales by adjusting the video frame rate, the DCT quantizer values, and the movement detection threshold. In *privilege quality* (PQ) mode, only the frame rate is adjusted. In *privilege rate* (PR) mode, only quantizer values and movement detection thresholds are adjusted [14]. IVS adjusts the target bit rate to network congestion by using a multiplicative backoff/additive gain algorithm very similar to that used for flow control in TCP [14, 54]. Congestion detection is solely based on packet loss [14] and so does not account for effects of latency or "hints" about congestion from latency changes.

As mentioned earlier, the Tenet group has also produced a best-effort conferencing system for Ethernet networks, which are not supported in the reservation-based Tenet suites [18]. This system uses quadtree encoding to dynamically scale video in reaction to congestion. Audio is not scaled. Scaling is augmented with a display buffering scheme at the receiver. Much of the success of the Tenet best-effort scheme appears due to the display buffering rather than the video scaling since without display buffering the system experiences wide swings in frame rate [18].

Kanakia, *et al.* results show feedback-based techniques can produce good quality with graceful degradation in congested networks [64]. Their scheme scaled the video stream using the DCT quantization factor and used small, fixed-size packets to transmit the stream. In this scheme, intermediate routers add queue lengths and service rates to each packet as it passes through the router. The scheme attempts to match the queue length at the bottleneck server (*i.e.*, the router with the lowest service rate) to a target value using gain equations. Since the service rates and buffer occupancy change during the feedback interval, the system estimates the values from the last known values. They simulated congestion by introducing cross traffic through the routers and by reducing the link transmission rates. They used a Mean Opinion Score (MOS) with small number of people to measure perceptual quality of delivered

video. The people judged video quality acceptable even during the congested periods. Kanakia, *et al.* capture the spirit of the best-effort school as follows.

> "For now, we invoke Occam's Razor principle, namely, that a simple approach if proved adequate should be preferred over complex ones. We use this argument vis-a-vis those schemes that rely on scheduling mechanisms to order packet transmissions to reduce packet losses and late arrivals. These approaches are considerably more complex than the feedback based approach discussed here." [64]

All of the adaptive best-effort schemes discussed so far have relied upon scaling, primarily video scaling, to change the bit rate of the media streams in response to congestion. In contrast, Roussos, *et al.* [95] uses adaptations based on changing the packaging of audio frames at intermediate nodes (*i.e.*, bridges and routers). Each intermediate node adapts based on the number of filled buffers in the node. Each intermediate node maintains two queues, one for audio packets and one for data packets. When buffer occupancy reaches some threshold level, the intermediate node transmits *choke* packets back to the data sources. When a data source receives a choke packet, the source must reduce its data transmission rate, but not its audio transmission rate. The intermediate nodes service the audio queue on a priority basis and do not throttle audio. During extremely congested periods, the data queue may not be serviced at all. The audio queue has minimum and maximum packet occupancy numbers, P_{min} and P_{max}, respectively. The intermediate node will not transmit an audio packet until at least P_{min} packets are available for transmission, thus an outgoing packet may carry more than one audio packet. If the intermediate node is unable to transmit the queued audio packets before the number of audio packets in the queue reaches P_{max}, new audio packets arriving on the inbound link replace the oldest audio packets already in the queue. This scheme adaptively matches the audio packet transmission rate at the intermediate nodes to the rates supported by the outbound network link. Unfortunately, this scheme requires changes to existing intermediate nodes and to the data sources. Furthermore, the scheme does not address video transmission.

The visual audio tool (vat) [53] and the NEtwork VOice Terminal (NEVOT) [96] are two other audio-only systems. Both systems support a variety of audio compression schemes, with bit rates in the 4.8 Kbits/second to 64 Kbits/second range, and use one of several existing transport protocols. The vat system manages network jitter by

using a target display latency to build a display queue. NEVOT also uses a display queue to manage jitter and augments this with a strategy of inserting or deleting silence periods at playback to manage the buffer queue length. NEVOT detects silence periods and only transmits talkspurts. NEVOT also allows a user-specified packetization factor that determines the number of audio frames packed into a given network packet. Like CU-SeeMe, NEVOT uses a reflector to support multicast audio conferences.

Although most rate-based schemes are integrated with resource reservation schemes, a few rate-based schemes have been proposed for networks without reservations. Haas describes a general framework for rate-based control schemes that uses continuous feedback to adapt a generic measure of network load [43]. The scheme uses the network load value with an adaptive gain/loss equation to control the transmission rate of the stream. Provided with a mechanism for scaling video to match the desired output rate, this framework becomes a generic video scaling transmission scheme. The Adaptive Rate Based (ARB) algorithm in APPN [47] is similar to Haas' framework and could be used as the basis of a video scaling algorithm provided some mechanism exists for ARB to communicate the desired transmission rate back to the video conferencing application.

2.4. Conclusion

The size of continuous media data generally forces the video conferencing system to compress the data before transmitting it across the network. Fortunately, continuous media is highly redundant and can be significantly compressed using lossy techniques with little perceptual effect. Many compression techniques have been developed to compress both audio and video. Unfortunately, although compressing the media streams makes transmission of the data easier, it does not solve the problem of transmitting the real-time stream across a network. Schemes to handle transmission of continuous media fall into two general categories: (1) resource reservation techniques and (2) best-effort techniques.

Resource reservation schemes can deliver high quality video conferences, but most existing computer networks do not support resource reservations due to the complexity and expense associated with reserving network resources. Since we are primarily interested in supporting conferences on existing computer networks, we must rely on best-effort transmission control of audio and video.

Most existing best-effort video conferencing systems either do not adapt to network congestion or rely on video scaling techniques to reduce the bit rate of the video stream when the network is congested. Many video scaling techniques have been developed and most exploit some aspect of the media compression algorithms to change the media stream bit rate. In many situations scaling is an effective response to network congestion, but as we will demonstrate in the next chapter, scaling is not always the best way to adapt to congestion. We have developed a transmission control framework that is more complete and robust than existing best-effort schemes. Our scheme uses both media scaling and media packaging to adapt transmission of the media streams to the current network conditions. The following chapter describes this framework in detail and later chapters demonstrate that high quality conferences can be sustained using best-effort techniques even on heavily congested networks.

Chapter III
Transmission Control Framework

This chapter introduces the transmission control framework. The framework has two purposes. First, the framework describes the capabilities of a video conferencing system as a set of operating points. Second, the framework relates these operating points to human perception and network congestion. This chapter also discusses the nature of network congestion and the types of congestion that may be present in a network. We introduce a simple queueing model to motivate our characterization of network congestion and relate this queueing model to the transmission control framework. Using the framework, we show why we need different adaptations to ameliorate the effects of different types of congestion. The framework becomes the basis for an adaptive transmission control algorithm that is described in the following chapter.

3.1. Characterizing a Video Conference System

Figure 3-1 shows the architecture of a video conferencing application. Abstractly, a video conferencing application is a program that generates and receives time-ordered sequences of audio and video samples called *frames*. Let the bit rates at which audio and video are generated be b_a bits/second and b_v bits/second, respectively, and let f_a and f_v be the corresponding frame rates (frames/second). We assume frames are generated periodically: one audio frame is generated every $1/f_a$ seconds and one video frame is generated every $1/f_v$ seconds.

To simplify the discussion we consider media transmission in only one direction. Two-way media traffic is handled as independent streams. In this chapter, audio frames are never transmitted in the same messages as video frames, but a single message may carry multiple frames from the same stream. There is no fundamental reason that audio and video cannot be packaged together, but it is simpler to explain the framework with independent media streams. All equations, relations, and algorithms in this dissertation treat each media stream of the conference independently unless otherwise noted. The transmission control algorithms described here control

the conference media streams independently. Although independent control turns out to work well, as demonstrated in the following chapters, it is also reasonable to consider controlling all the streams of the conference in concert.

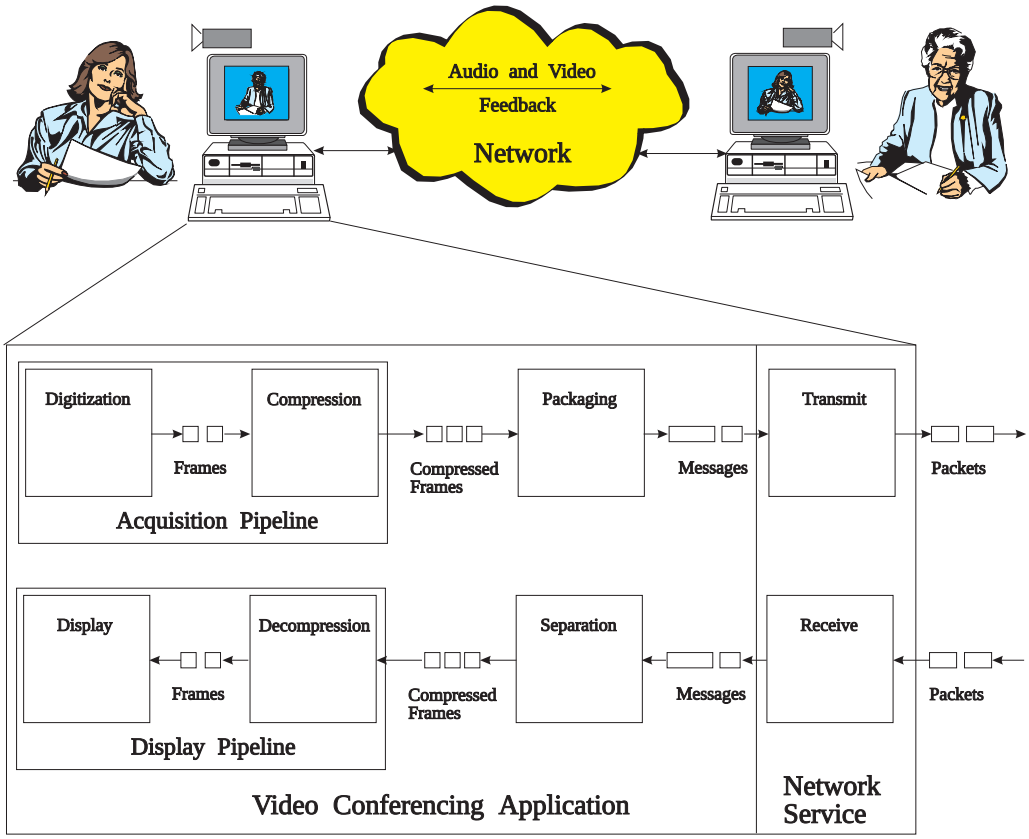


Figure 3-1: Video Conference Application Architecture

3.1.1. Operating Points

At well-defined points in time, the conferencing application chooses a message rate and a bit rate for each media stream. The application may change the message and bit rates during the course of the conference. The application may change the bit rate by changing either the frame size (*e.g.*, by changing the coding or compression scheme) or the frame rate (*e.g.*, by temporal scaling [26]). The application may change the message rate by varying the number of frames packed into a single message. Conceptually, we think of the *Digitization* and *Compression* stages in Figure 3-1 as having "knobs" that the application may adjust to set the size and generation rate of

frames and the amount of compression done on the frames. The *Packaging* stage also has a conceptual knob that the application can set to control the number of media frames packaged into a message.

The allowable settings for the knobs determine the properties of a particular audio/video conferencing system. We characterize each media stream *s* in a conference by the set OP_s of *operating points* in a message rate × bit rate space. For stream *s*, $(m_s, b_s) \in OP_s$ if and only if the conference is capable of generating b_s bits/second and partitioning *s* into m_s messages/second. Figure 3-2 shows the set of operating points for the video stream of one of the conferencing systems used in this work. This system has a choice of three video coding schemes; a *high* quality scheme (approximately 8000 bytes/frame), a *medium* quality scheme (approximately 4000 bytes/frame), and a *low* quality scheme (approximately 2000 bytes/frame). Each scheme generates at most 30 frames per second. This particular system always sends each video frame in a separate message. Coding is frame independent, so the system may transmit video at any message (frame) rate from 1-30 messages per second. For example, the conference may transmit 30 messages/second during part of the conference, but only 15 messages/second at other times. Thus the set of video operating points contains 90 points, 30 per coding scheme (see Figure 3-2).

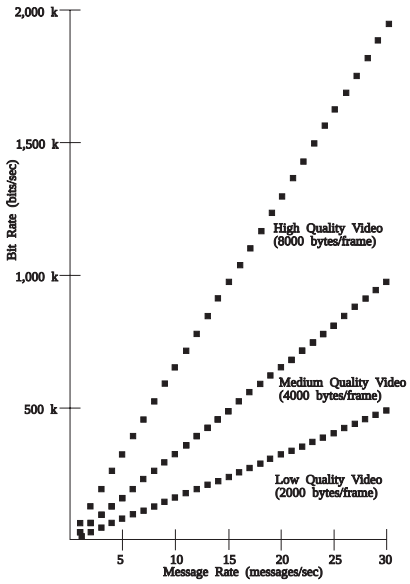


Figure 3-2: Video Operating Points

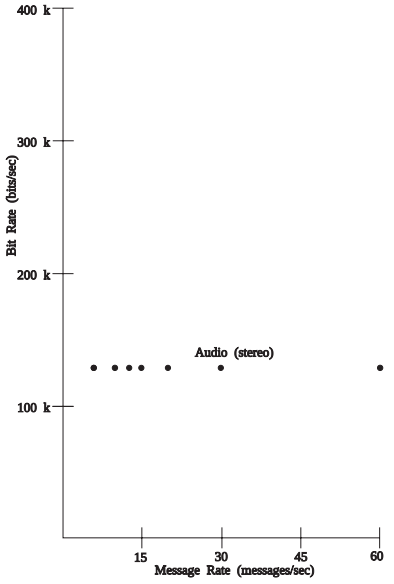


Figure 3-3: Audio Operating Points

Figure 3-3 shows the audio operating points for the system. The system is capable of generating only 1 bit-rate. The system generates sixty 250 byte audio frames each second and can transmit 60, 30, 20, 15, 12, 10, or 6 messages per second (corresponding to 1, 2, 3, 4, 5, 6, or 10 audio frames per message). The circles in Figure 3-3 represent the audio operating points. It is sometimes convenient to represent the operating points for all media streams in a single graph, as illustrated in Figure 3-4.

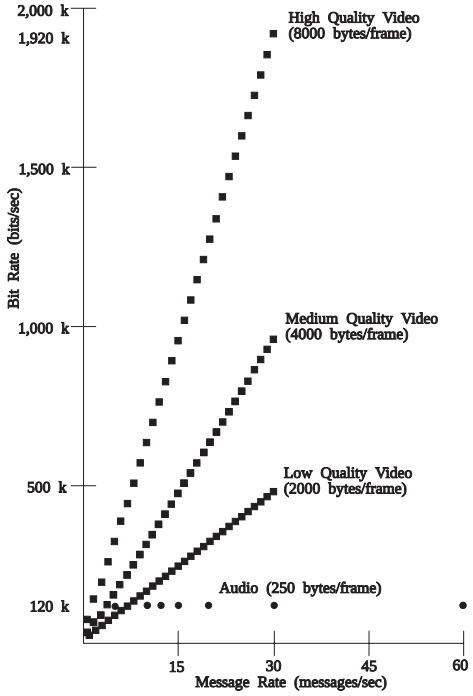


Figure 3-4: Operating Points (video as squares; audio as circles)

For each media stream, the application selects an operating point and then produces the stream at the rates specified by the operating point. Over time, the application may alter its operating points. For example, at some time the application may choose to generate 60 audio frames per second and send each frame in a separate message (*i.e.*, the conference is operating at (60, 120k) in Figure 3-3), but may later choose a different operating point (*e.g.*, (30, 120k)). Transmission control is the process of determining when to change the operating point and which new operating point to select. There is only a finite, though perhaps large, set of operating points for any digital system, so OP_S completely characterizes the conferencing system's options for generating and transmitting media stream *s*. An implicit assumption of transmission control is that at least one operating point for each media stream provides adequate

fidelity and latency for a successful conference and can be successfully delivered in the current network environment. If no operating point satisfies these two requirements, it is impossible for any transmission control scheme to succeed.

If each media stream is transmitted independently, there is no technical reason to show the operating points for each stream on the same graph since there is no relationship between the operating points of the two streams. If the conferencing application may transmit audio and video frames in the same message or the bit rates of the two streams are in some way interdependent, there is logically only a single, combined media stream and a single set of operating points represents the combined media stream. For example, consider a system that generates 60 audio frames/second and 30 video frames/second, but always transmits two audio frames with an associated video frame. In this system, there is only one combined media stream and a single set of operating points, as shown in Figure 3-5. The message rate may vary from 1 to 30 messages per second and the bit rate associated with a particular message rate is the sum of the audio and video bit rates. This example shows how multiple media streams may be represented and controlled in concert. During later discussions, we discuss constraint relationships and control algorithms in terms of an individual stream. The reader should keep in mind that the technique described here of representing the entire set of streams as a single aggregate stream logically creates a single stream. The relations and algorithms described later can be directly applied to the aggregate stream.

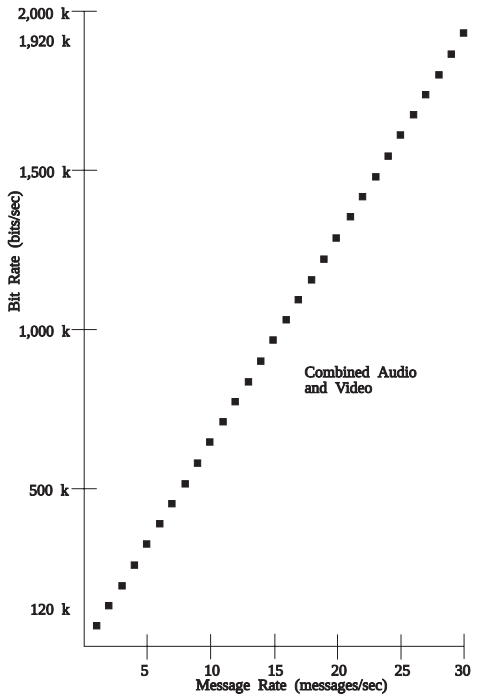


Figure 3-5: Combined Audio/Video Stream Operating Points

As demonstrated by Figures 3-4 and 3-5, the set of media operating points are system specific; however, the operating points are always represented in a message rate × bit rate space and the transmission control framework discussed in this chapter is system independent. We describe several hypothetical sets of operating points in the next section to show how operating points define the capabilities of a conferencing system.

3.1.2. Examples of Operating Points for Other Systems

Figure 3-6 shows a video stream with frame independent coding. Since frames are independent, the system may potentially generate any frame rate between 1 and 30 frames/second. Only one video coding scheme is available and it produces approximately 4000 bytes per video frame. This video conferencing system may transmit as many as three video frames per message, but the system only generates a frame rate that is a multiple of the number of frames packed per message. For example, the operating points at (10, 1,000k) and (30, 1,000k) generate the same frame rate (30 frames/second) and the same bit rate (1,000k bits/second), but with different message rates. The operating point (10, 1,000k) packages 3 frames/message, so the message rate is 10 messages/second. The operating point (30, 1,000k) puts each frame in a separate message, so the message rate is 30 messages/second.

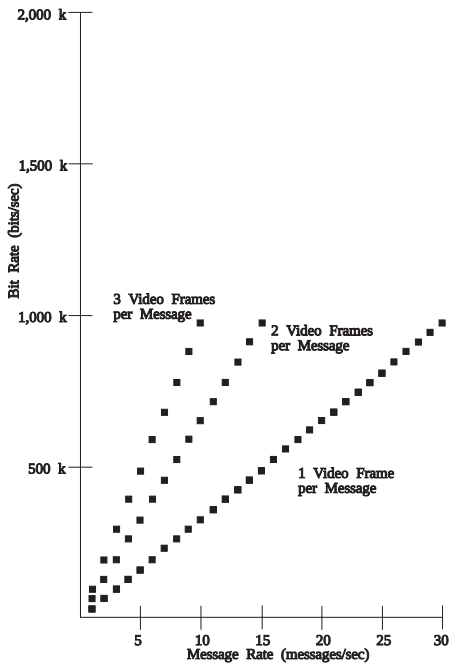
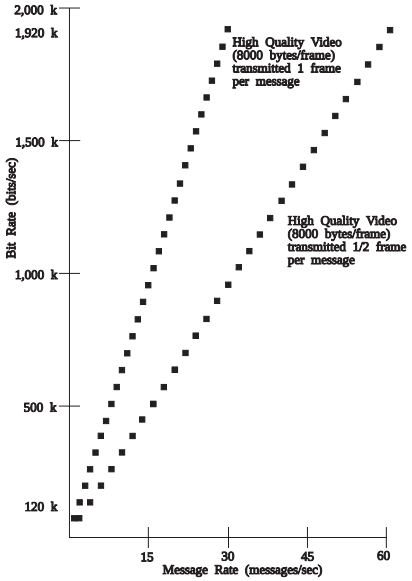


Figure 3-6: Frame Independent Video Operating Points

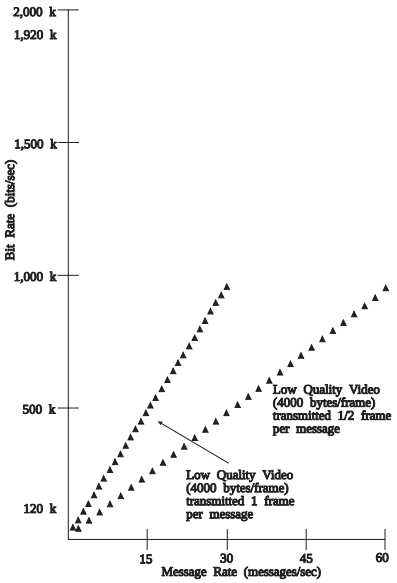
The system described in Figure 3-7 also uses frame-independent coding, but has more generation and transmission options and thus more operating points. This system can generate video using two different video coding schemes. The high quality video coding scheme generates an average of 8000 bytes per frame. The low quality video coding generates an average of 4000 bytes per frame. The system generates up to 30 frames per second for either coding scheme. The system can transmit a single video frame as either one or two messages (*e.g.*, by transmitting a low resolution image first, the transmitting a second scan to enhance the image), so it has a maximum message rate of 60 frames per second.

Figure 3-7 (a) shows the operating points for the high quality video encoding scheme. There is an operating point for every message rate between 1 and 30 messages per second. These operating points represent sending a complete frame in a single message for every possible frame rate. There is also an operating point for every even message rate from 2 to 60 messages/second. These operating points result from sending two messages for every generated frame. Figure 3-7 (b) shows the operating points for the low quality video encoding scheme. Like the high quality encoding scheme, each possible frame rate results in two operating points. One operating point

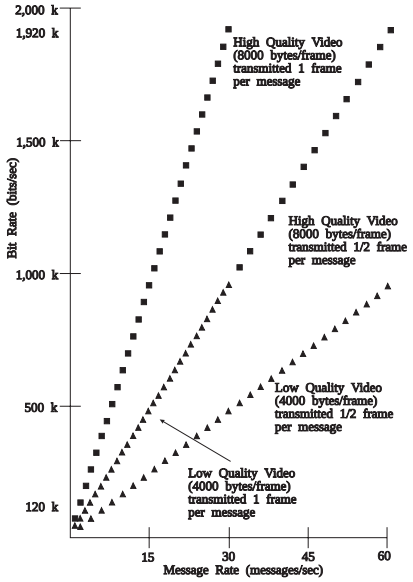
represents sending the entire frame in a single message and the other operating point represents sending the frame in two messages.



(a) High Quality Video Operating Points



(b) Low Quality Video Operating Points



(c) Combined High and Low Quality Video Operating Points

Figure 3-7: Multiple Coding Frame Independent Video Operating Points

Since there are fewer bits per frame with the low quality encoding, the bit rates associated with the low quality operating points are generally lower than those of the high quality operating points. However, there are situations when the low quality encoding generates exactly the same message and bit rates as the high quality encoding. In particular, for all the even numbered message rates between 2 and 30 messages/second, the low quality video scheme transmitted one message/frame generates the same message and bit rates as the high quality scheme transmitted at two messages/frame. For example, when the system transmits low quality video using one message per frame, the operating point for two frames/second is (2, 8k). When the system transmits high quality video using two messages per frame, the operating point for one frame/second is also (2, 8k). This situation occurs for all even message rates up to 30 messages per second. For example, the operating point for low quality video generated at 30 frames per second and transmitted one frame per message is (30, 120k). The operating point for high quality video generated at 15 frames per second and transmitted at two messages per frame is also (30, 120k).

From a transmission control perspective, operating points with the same message and bit rates are indistinguishable. The operating points generate the same number of messages and the messages are roughly the same size, so the operating points have the same impact on the network. Since the transmission control scheme cannot distinguish between the operating points, it does not make sense to describe more than one operating point to the transmission control scheme for a given message and bit rate combination. In this case, the designer of the video conferencing system must decide *a priori* which coding scheme (*i.e.*, the high quality or low quality video) to define to the transmission control framework for a given operating point. The system designer should make this decision based on the perceptual quality of the two schemes at the specified operating point and choose whichever is better. Figure 3-7 (c) shows the combined set of operating points for the high and low quality video encodings. In this case, the system designer has chosen to use the lower quality video at a higher frame rate when there are two possible implementations of an operating point.

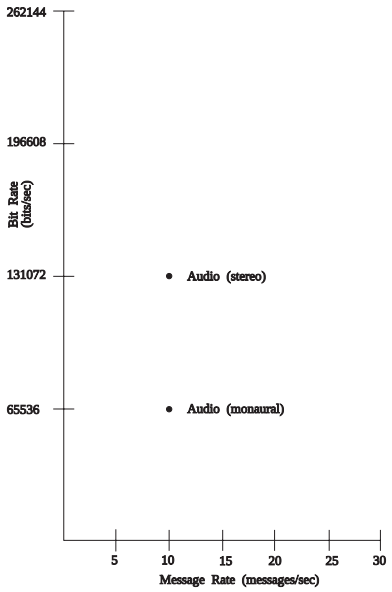


Figure 3-8: Monaural/Stereo Audio Operating Points

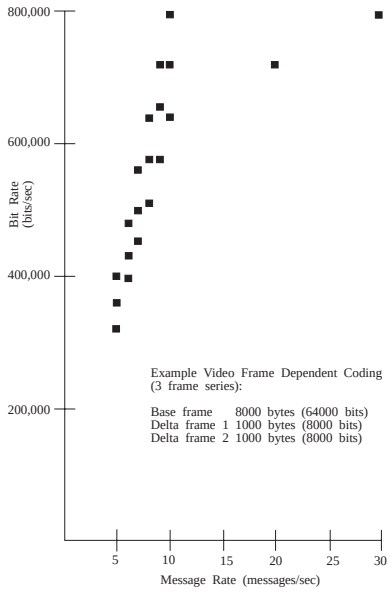


Figure 3-9: Frame Dependent Video Operating Points

Figure 3-8 shows the operating points for an audio stream capable of transmitting either monaural or stereo audio by sending either one or two audio channels. Each channel has a bit rate of 64 kbits/second. The conferencing application collects audio in 10 millisecond samples, but transmits exactly one message every 100 milliseconds. Each message carries 10 samples per transmitted channel and there are 10 messages transmitted each second. This system only supports two combinations of message and bit rates. The entire set of potential operating points consists of only two points corresponding to sending either one or both channels. In this example, the relationship between sampling rate and message rate is not apparent in the operating point description of the system. Since the system is capable of transmitting only 10 messages/second, from a transmission control perspective, the actual sampling rate is irrelevant. This example illustrates how the definition of operating points omits details beyond the control of the transmission control algorithm and provides the essential abstraction of the media stream from a transmission perspective.

Figure 3-9 shows the operating points for a video stream using frame dependent coding. This system generates a full-image base frame followed by two delta frames that represent only the changes between the base frame and later frames. The conferencing application may send each base and delta frame as a separate message or a message may carry a base frame plus one or two delta frames. For example, the

operating point (30, 800k) corresponds to sending all generated frames, both base and delta frames, as individual messages (*i.e.*, 10 base frames/second + 20 delta frames/second = 30 frames/second, which implies 30 messages/second since each frame is carried by one message). The operating point (20, 720k) corresponds to sending each base frame and one of the two delta frames as individual messages. In this case, the second delta frame is never transmitted. The remaining operating points represent sending a base frame together with 0, 1, or 2 delta frames in each message.

Figure 3-9 shows how potential combinations of message and bit rate that are technically possible with a conferencing system may be omitted as operating points. These points are useless either because they cannot lead to successful display of the media stream or because they provide quality that is inadequate for the conferencing application. In other words, the formulation of the set of operating points may contain implicit quality assumptions. In this example, the system designer has determined that sending less than five base frames per second leads to unacceptable video quality, so there are no operating points below five messages per second. Also, although the system is capable of transmitting only the delta frames without the base frames, for a potential operating point of (20, 160k), the receiver cannot play the delta frames without the base frame, so (20, 160k) can be removed from the operating point description *a priori*. Obviously, the individual defining the set of operating points for a particular conferencing system must know a great deal about the capabilities of the system and about the viability of the operating points technically achievable with the system. The pruning of obviously bad, but technically possible, operating points is our first example of constraints on the viable operating points. Not all operating points are true candidates for use with the video conference. Some of these infeasible operating points may be excluded *a priori*, but others are only excluded at certain times, such as during periods with heavy network congestion. Much of the rest of this chapter deals with pruning some operating points from consideration based on properties of the current network environment.

3.1.3. Fragmentation and Operating Points

The operating points described in the previous section are entirely derived from the capabilities of particular video conferencing systems. The operating points are defined without regard to the type of network used to carry the conference. Message rates are defined independently from the amount of data carried by the actual network

packets (*i.e.*, the size of the packet *payload*). However, the size of the packet payload may have a major impact on the effect of network congestion upon the conference, as discussed later in this chapter. This section describes the relationship between the operating points characterizing the system and the size of the packet payloads in a network. In particular, this section describes how we can calculate the *realization* of operating point (m_s, b_s) for each hop in the network path.

Let a network *node* be the source or destination computer for a conference or a network interconnection point such as a router or a bridge. A *link* is a network segment connecting two nodes. A *hop* is composed of a network adapter, the associated transmission link, and the following network node. A *network path* is composed of a series of hops (see Figure 3-10). The realization of an operating point is a mapping from the operating point to a generated packet rate at a hop.

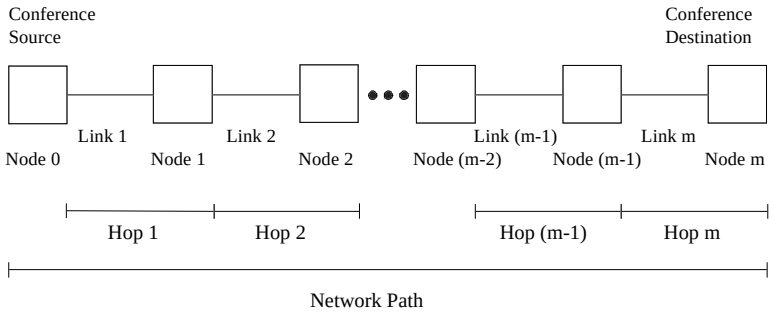


Figure 3-10: Network Nodes, Hops, and Paths

If the size of the packet payload for each link along the network path is greater than the largest application message size, the stream message rate and the resulting packet rate are the same for all hops in the path (*i.e.*, $m_s = p_{s,k}$ for each hop k, where $p_{s,k}$ is the generated packet rate for stream s on hop k). However, we must calculate the packet rates for paths that have one or more hops where the size of the packet payload is smaller than the largest message size. If the message is larger than the maximum payload for a hop, the message must be fragmented into smaller packets before it is transmitted across the hop. We assume messages are not reassembled until the destination. Since fragmentation is deterministic, we can create an algorithm that calculates the packet rate resulting from a particular operating point (m_s, b_s) for each hop. For concreteness, we describe this algorithm using the Internet Protocol (IP) as the underlying network protocol. Similar algorithms can be derived for any

fragmenting protocol. For simplicity, we assume all packets follow a single, static path and that no packets are lost, duplicated, or delivered out of order.

Before we can calculate the packet rates, we need to know some facts about our environment. MTU_k is the maximum logical transmission size in bytes for hop k. This is the maximum size of a packet including all protocol headers and user data. $Load_k$ is the maximum payload for hop k (*i.e.*, MTU_k minus the number of bytes required by the transport protocol to form a valid packet).

Listing 3-1 gives the algorithm for calculating the packet rates for an operating point (m_s, b_s). In Listing 3-1, m_s is represented by the variable m passed to the *Realize* subroutine, b_s is represented by the variable b, the number of hops in the path is represented by the variable h, and $Load_k$ for all hops k in the path is represented by the *load* array and is indexed from 1 to h. At each hop, we (conceptually) number each packet generated as a result of a message in the order in which they are generated. For example, the first packet generated on hop 3 resulting from a particular message is numbered 1, the next 2, *etc.*

We want to calculate (1) the number of packets generated on each hop as a result of operating point (m_s, b_s), (2) the size of each of these packets, and (3) the packet rate generated on hop k. In the algorithm, the number of packets generated on each hop for each message is held in the *packetcount* array. The index to the array corresponds to the hop number and ranges from 0 (*i.e.*, the conference source) to h. The *packetsize* array holds the size of each packet generated on each hop. The first index to the array identifies the hop and ranges from 0 (*i.e.*, the source) to h. The second index to the array identifies the conceptual packet number. All packets generated on hop k as a result of a particular packet from hop $(k - 1)$ carry $Load_k$ bytes, except the last packet. The last packet carries at most $Load_k$ bytes, but many carry fewer if there are too few remaining bytes to completely fill the packet. This calculation is handled by the **if-then-else** structure in the innermost loop. The generated packet rate is computed by multiplying the number of packets generated for each message on a particular hop k with the message rate m_s (*i.e.*, variable m). The algorithm operates by calculating the number of generated packets and the size of each of these packets starting with hop 1 and proceeding to hop h.

```
Realize(m, b, h, load[1..h])

packetsize[0, 1] ← b ÷ (m × 8)
packetcount[0] ← 1

for k ← 1 to h
begin
   print "Computing values for hop" k "with a maximum payload of" load[k]
"bytes."

   last ← k – 1
   packetcount[k] ← 0
   for i ← 1 to packetcount[last]
   begin
      packets[k, i] ← ceiling(packetsize[last, i] ÷ load[k])

      print "Packet" i "from hop" k "generates" packets[k, i] "packet(s) on hop" k"."

      for j ← 1 to packets[k, i]
      begin
         packet ← packetcount[k] + j

         if (j < packets[k, i]) then packetsize[k, packet] ← load[k]
         else packetsize[k, packet] ← packetsize[last, i] – ((j – 1) × load[k])

         print "Packet" packet "is" packetsize[k, packet] "bytes."
      end

      packetcount[k] ← packetcount[k] + packets[k, i]
   end

   print "Total packets for hop" k "is" packetcount[k]"."
   print "The generated packet rate for hop" k "is" m × packetcount[k]
         "packets/second."
end

return
```

Listing 3-1: Algorithm for Calculating Realizations

Using the algorithm in Listing 3-1, we can calculate the number of packets generated on each hop by one message and calculate the packet rate on each hop. For example, Figure 3-11 below shows a four hop network. In this network, $Load_1$ = 17,800 bytes, $Load_2$ = 1,500 bytes, $Load_3$ = 550 bytes, and $Load_4$ = 1,500 bytes. Suppose Node 0

sends video frames to Node 4. The operating point for video is (30, 960k). Each video frame is 4,000 bytes. Thirty video frames are generated every second and each video frame is carried in a single message. Listing 3-2 shows the output for this operating point from an implementation of the algorithm in Listing 3-1. A single message from the operating point (30, 960k) generates 1, 3, 8, and 8 packets on hops 1 through 4 in Figure 3-11, respectively. The packet rates for (30, 960k) are 30, 90, 240, and 240 packets/second on hops 1 through 4, respectively. The size of the packets vary between packets and hops.

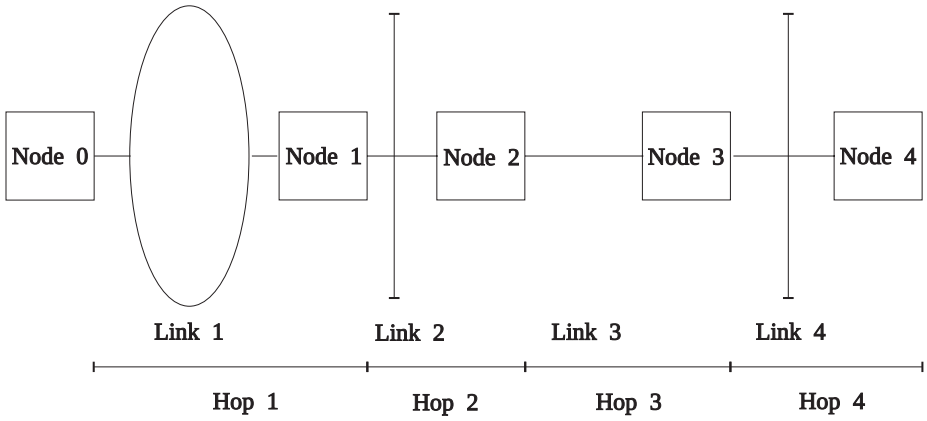


Figure 3-11: A Sample Multi-hop Network

We can run the algorithm in Listing 3-1 for each operating point $(m_s, b_s) \in OP_s$ to calculate the associated packet rate for each hop. This association is represented with a triple, $(m_s, b_s, p_{s,k})$. We call the set of triples resulting from the operating points in OP_s the *realization* of the operating points at hop k along the path. Although the realizations may be different at every hop, the application still may manipulate only the message and bit rates at the source (*i.e.*, the application may still only select operating points from OP_s). Fragmentation does not add any new operating points to OP_s. However, the effect of operating point changes at the source may impact hops differently (*e.g.*, generate different packet rates).

```
Computing the realization for operating point ( 30 , 960000 ).
Computing values for hop 1 with a maximum payload of 17800 bytes.
Packet 1 from hop 0 generates 1 packet(s) on hop 1.
Packet 1 is 4000 bytes.
Total packets for hop 1 is 1.
The generated packet rate for hop k is 30 packets/second.
Computing values for hop 2 with a maximum payload of 1500 bytes.
Packet 1 from hop 1 generates 3 packet(s) on hop 2.
Packet 1 is 1500 bytes.
Packet 2 is 1500 bytes.
Packet 3 is 1000 bytes.
Total packets for hop 2 is 3.
The generated packet rate for hop k is 90 packets/second.
Computing values for hop 3 with a maximum payload of 550 bytes.
Packet 1 from hop 2 generates 3 packet(s) on hop 3.
Packet 1 is 550 bytes.
Packet 2 is 550 bytes.
Packet 3 is 400 bytes.
Packet 2 from hop 2 generates 3 packet(s) on hop 3.
Packet 4 is 550 bytes.
Packet 5 is 550 bytes.
Packet 6 is 400 bytes.
Packet 3 from hop 2 generates 2 packet(s) on hop 3.
Packet 7 is 550 bytes.
Packet 8 is 450 bytes.
Total packets for hop 3 is 8.
The generated packet rate for hop k is 240 packets/second.
Computing values for hop 4 with a maximum payload of 1500 bytes.
Packet 1 from hop 3 generates 1 packet(s) on hop 4.
Packet 1 is 550 bytes.
Packet 2 from hop 3 generates 1 packet(s) on hop 4.
Packet 2 is 550 bytes.
Packet 3 from hop 3 generates 1 packet(s) on hop 4.
Packet 3 is 400 bytes.
Packet 4 from hop 3 generates 1 packet(s) on hop 4.
Packet 4 is 550 bytes.
Packet 5 from hop 3 generates 1 packet(s) on hop 4.
Packet 5 is 550 bytes.
Packet 6 from hop 3 generates 1 packet(s) on hop 4.
Packet 6 is 400 bytes.
Packet 7 from hop 3 generates 1 packet(s) on hop 4.
Packet 7 is 550 bytes.
Packet 8 from hop 3 generates 1 packet(s) on hop 4.
Packet 8 is 450 bytes.
Total packets for hop 4 is 8.
The generated packet rate for hop k is 240 packets/second.
```

Listing 3-2: Output of Realization Algorithm for Operating Point (30, 960k) on the Network in Figure 3-11

As an example, consider the network in Figure 3-12 and the set of audio and video operating points in Figure 3-4. The MTU of the first 16 megabit/second token ring segment in Figure 3-12 is 17,800 bytes and is sufficient to carry any of this particular application's messages in a single packet; therefore, the message and packet rates on the first hop are the same. However, the Ethernet MTU (1,500 bytes) is insufficient to carry many of the application messages in a single packet, so the message and packet rates are no longer the same on the second hop. Since IP does not reassemble at intermediate nodes, the relationship between messages and packets is the same on the second token ring segment as on the Ethernet segment. Figure 3-13 shows the three-dimensional representation of the realization of the operating points for the Ethernet and second token ring hops. To make the figure easier to read, we have represented the realization of the operating points as columns rather than points in a three-dimensional space (*i.e.*, the point (x, y, z) is represented as a column from (x, y, 0) to (x, y, z)). We do not show the realization for the first token ring segment, since the message and packet rates are the same.

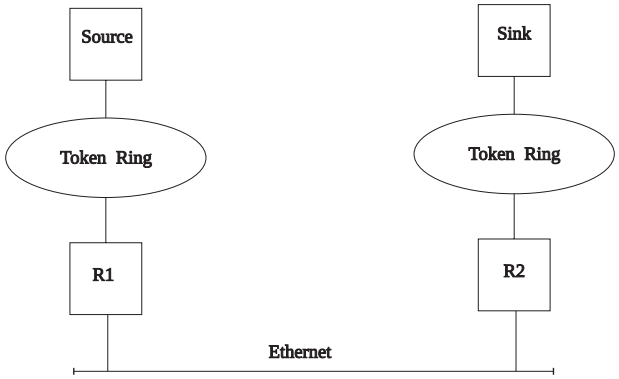


Figure 3-12: Ethernet Backbone Network

Figure 3-13 shows the large increase in packets on the Ethernet and second token ring resulting from the fragmentation of video messages. On the first token ring segment, the maximum message rates for audio yields 60 packets per second and video yields 30 packets per second (Figure 3-4 and $m_a = p_{a,1}$, $m_v = p_{v,1}$). On the Ethernet and the second token ring (Figure 3-13), video has a maximum packet rate of 180 packets per second since 6 Ethernet packets are needed to carry each 8,000 byte video frame and there are 30 frames per second. The maximum audio rate remains 60 packets/second since the maximum message rate is 60 messages/second, each message contains one audio frame, and an audio frame is small enough to fit into one Ethernet packet.

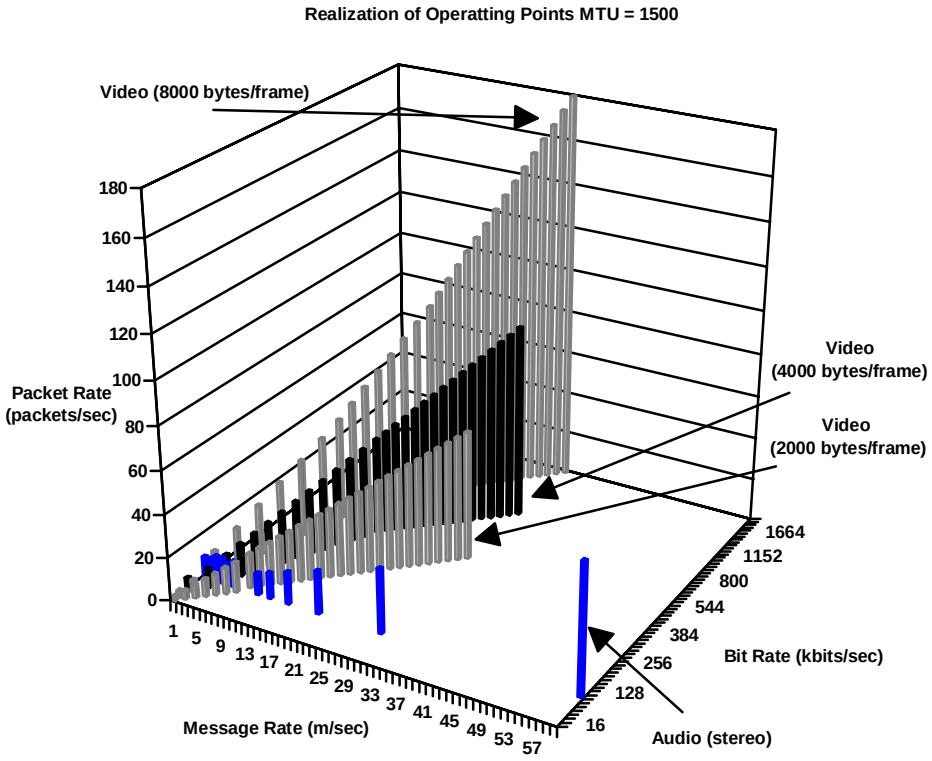


Figure 3-13: Realization of Operating Points on Ethernet

Figure 3-14 is an enlarged view of the audio points from Figure 3-13. Unlike video, the packet rates associated with audio are not monotonically increasing. In this example, decreasing the message rate sometimes increases the number of packets on a hop. This phenomenon occurs when the packaging reaches the point where the n^{th} frame is successfully packed into a packet, but frame $n + 1$ causes the message size to exceed the packet's capacity and require an additional packet. This situation occurs whenever $n + 1$ frames are packaged in a single message. In this particular example, the video packet rate is monotonically increasing since all video frames exceed the 1,500 byte Ethernet MTU (or more accurately, the maximum packet payload for the Ethernet) and each message carries a single frame (see Figure 3-13). However, depending on the conferencing system and the network technologies involved, video may also exhibit non-monotonic behavior.

Even though the generated packets on a hop may not always decrease with decreases in the message rate, as message size increases, the packaging of frames into packets becomes increasingly efficient and the packet rate generally decreases. This leads us

to the hypothesis that manipulating the message rate at the conference source can effectively control the generation of packets anywhere along the transmission path. We claim that this manipulation of message rate allows the conference source to adapt to congestion along the network path even if the messages are fragmented. If our claim is true, then a transmission control algorithm can successfully manage the transmission of the media streams solely through careful selection of operating points. The control algorithm need not know the physical characteristics of the network. This is a particularly important point because it is often difficult or impossible for end-to-end transmission control schemes to discover the network topologies and MTU sizes for the entire path.

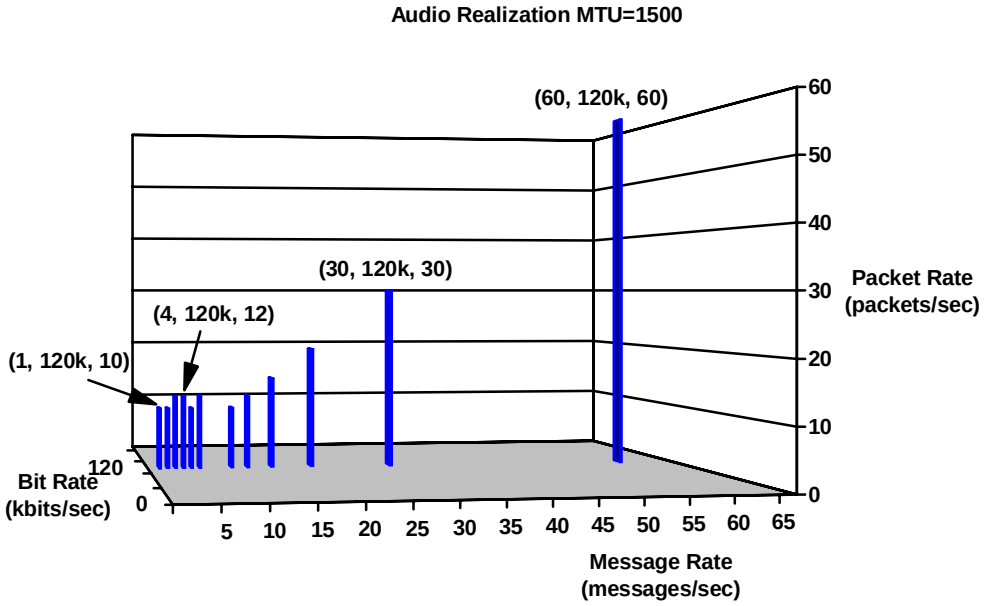


Figure 3-14: Audio Packet × Message Rates (MTU =1500)

3.2. Perceptual Constraints

The key problem for transmission control is to choose a *feasible operating point*. For stream *s*, an operating point is *feasible* if and only if it (1) provides acceptable latency and fidelity for stream *s* and (2) it is sustainable given the current level of congestion in the network. We call the constraints resulting from the fidelity or latency considerations *perceptual constraints* because of their dependency upon human perception. We call the constraints resulting from the network *transmission constraints*. This section discusses the perceptual constraints and the following section discusses transmission constraints.

3.2.1. Fidelity Constraints

We define media fidelity as the degree to which the display of a media stream matches the captured media stream. We consider a conference to have perfect fidelity if the played media exactly matches the captured media. We do not consider the degree to which the captured media stream reflects the media source (*e.g.*, how well a recording captures live music), since this is a function of the hardware and software used to sample the source. We are concerned with how well the data provided to the conferencing application, given the limits imposed by the capture hardware and software, are delivered and played on the receiving end. The played media stream may differ from the captured stream in several ways. First, the captured frame rate may be much higher than that delivered or played. This may be due to frame loss during transmission or due to intentional transmission of only a subset of the captured frames (*e.g.*, through the use of temporal scaling). Second, the played bit rate may be much smaller than the captured bit rate. One way to reduce the bit rate is by reducing the delivered frame rate as described above. If the coding scheme is unchanged, delivering fewer frames also reduces the number of bits delivered. However, there are ways to reduce the bit rate that do not affect the frame rate. For example, only a portion of the DCT coefficients associated with a video image may be transmitted. In this case, the video frame rate stays the same, but the bit rate of the stream is reduced. For many audio and video compression schemes, subtle differences between captured and played frames are often not noticed by humans; however, there are clearly limits to our perceptual tolerance [86]. The relationship between bit rate, frame rate, and the perceived quality of the conference is complex and beyond the scope of this paper; however, the delivered media fidelity must be adequate to support the conference. We

assume that the individual responsible for identifying the operating points has knowledge of the bit and frame rate combinations required for adequate fidelity. We call the limits imposed by fidelity considerations *fidelity constraints*. We describe fidelity constraints by relating operating points to the minimum acceptable bit rates for each media stream.

Fortunately, fidelity constraints typically do not vary over time and are usually known *a priori*. Let F_s^{min} be the minimum bit rate required for acceptable display of stream *s*. For acceptable fidelity, an operating point (m_S, b_S) must satisfy the relation $b_s \geq F_s^{min}$. F_s^{min} may vary with different media streams and different applications. For example, consider a video conferencing application with the operating points shown in Figure 3-15. Suppose that for a "talking heads" conference, we determine that a video bit rate of 150,000 bits/second provides an acceptable lower bound on the required video fidelity. Thus operating points with bit rates below 150k should not be used. Said another way, the application is constrained in its choice of operating points by a fidelity constraint. This constraint is graphically represented by the area in Figure 3-15 labeled "Fidelity Constraint." In this example, the fidelity constraint eliminates only a few operating points from consideration and many candidate operating points remain from both the high and medium quality video encodings. Clearly, some of the operating points provide better relative fidelity than others, but most of the operating points provide acceptable fidelity as defined for the talking heads application. Now consider Figure 3-16, that shows the same video conferencing system, but with the fidelity constraints required for a medical diagnosis application. In the medical application, the requirements for video fidelity are much higher than with the previous application. In Figure 3-16, the fidelity constraints restrict the conferencing application to choosing only video operating points that use the high quality encoding and have high frame rates. All the medium quality video operating points are eliminated from consideration, as are the low frame rates for the high quality encoding.

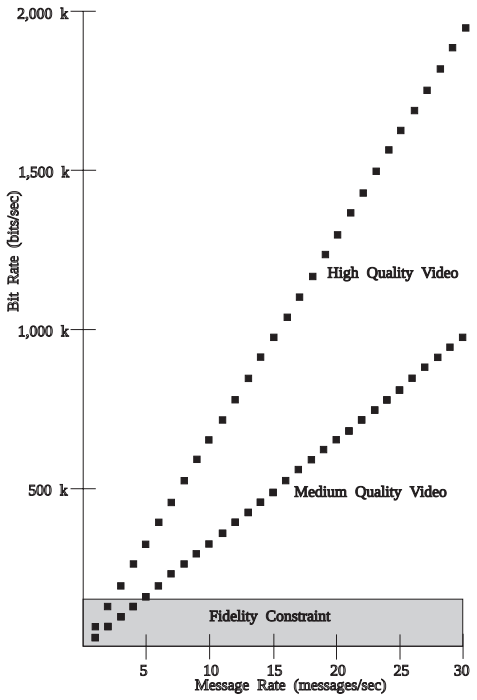


Figure 3-15: Sample Fidelity Constraints for "Talking Heads" Video

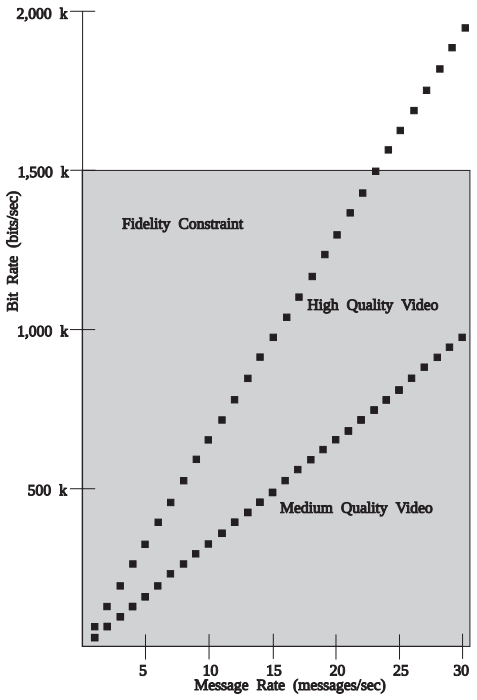


Figure 3-16: Sample Fidelity Constraints for Medical Diagnosis Video

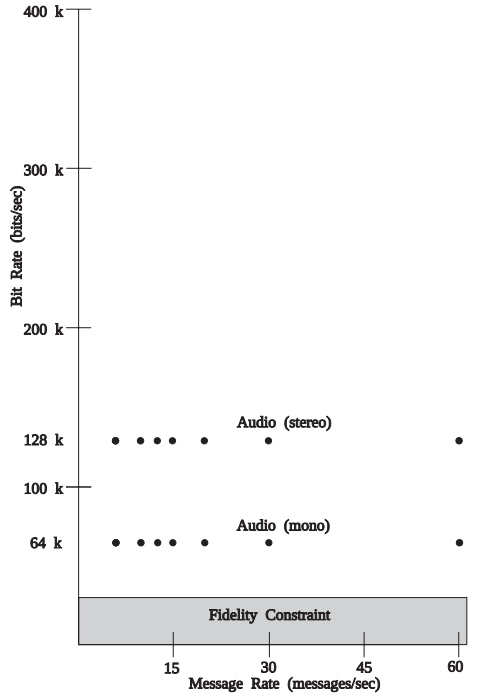


Figure 3-17: Sample Fidelity Constraints for "Talking Heads" Audio

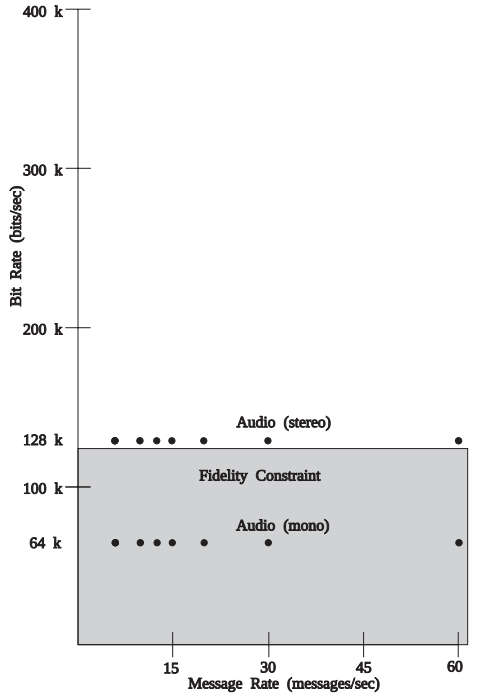


Figure 3-18: Sample Fidelity Constraints for Music Composition Audio

As another example, Figure 3-17 shows the audio operating points for a conferencing system. For the talking heads application, the bit rate for adequate audio fidelity is so low that no operating points are eliminated by the constraint. Figure 3-18 shows the same system with perceptual constraints for a distributed music composition application. In this case, the minimum audio bit rate excludes monaural playback (and correspondingly transmission of the monaural stream).

Obviously, the relation $b_S \geq F_s^{min}$ is an over-simplification of fidelity constraints. We may also eliminate specific message × bit rate combinations that provide unacceptable fidelity when constructing the set of operating points for a particular conferencing system. For example, recall the frame dependent coding example earlier. It does not make sense to include the message × bit rate points derived from sending only the delta frames of an inter-frame coding scheme (without the base frames), even though the system may be able to deliver such combinations. It is reasonable to prune all combinations of message and bit rates providing unacceptable fidelity from the definition of the system operating points since these points can never be used to provide an adequate quality conference. A set of operating point such as that shown in Figure 3-4 may have already captured many fidelity constraints for a particular conferencing system before the system is ever run. Obviously, only someone with knowledge of the particular coding schemes and the goals of the conferencing application can remove such points from consideration *a priori*. For our purposes here, the simple bit rate relation above suffices to illustrate the concept of fidelity constraints, with the recognition that many fidelity constraints may have been identified in the process of determining the operating points.

3.2.2. Latency Constraints

Human perception also limits the acceptable media latency. High latency interferes with conversational interaction and thus may make a set of operating points infeasible. Unlike the fidelity constraints, latency constraints vary over time and are not typically known *a priori*. We distinguish between several measurements of latency. *Network latency* is the difference between the time a message is available to the network service on the sending machine and the time the message is delivered to the separation stage at the receiver (see Figure 3-1). $L_S(t)$ is the average network latency for stream *s* at time *t*. The *transmission latency* for a particular frame *f* in stream *s* is the difference between the time *f* is available to the packaging stage at the sender and the time *f* is

delivered to the separation stage at the receiver. The difference between network latency and transmission latency is the amount of time a frame spends buffered on the sending side, which we call the *induced latency* for frame *f*. The *end-to-end latency* for a particular frame *f* in stream *s* is the difference between the time the message carrying *f* is delivered to the display stage at the receiver and the time *f* is available to the packaging function at the sender. Although end-to-end latency is the true measure of conference latency (at least from the participants' perspective), end-to-end latency is dependent on not only the induced and network latencies, but also on the display policies at the receiver [103]. The latter are outside the scope of this paper, so we will confine our discussion to transmission and network latency.

Let L_s^{max} be the maximum tolerable transmission latency for any frame *f* in stream *s*. Like F_s^{min}, L_s^{max} may vary by media stream or application. To ensure acceptable transmission latency, the sender may buffer *s* frames for at most $Buf_s^{max}(t) = \max(L_s^{\max} - L_s(t), 0)$ seconds before they must be transmitted. Therefore, at time *t* + 1, each stream *s* frame must be transmitted within $Buf_s^{max}(t)$ seconds and hence messages must be generated no slower than the rate of 1 every $\lceil f_s Buf_s^{max}(t) \rceil / f_s$ seconds (*i.e.*, a minimum message rate of $\left\lceil \frac{f_s}{\lceil f_s Buf_s^{max}(t) \rceil} \right\rceil$ messages/second).

For example, supposed the conference audio stream has the operating points in Figure 3-19 and let the maximum acceptable latency for audio be 250 *ms* (L_a^{max}= 250 *ms*). The system generates 60 audio frames per second (f_a = 60 frames/second). Assume at time *t* the network latency for audio, L_a(t), is 100 *ms*, so $Buf_a^{max}(t)$ = 150 *ms*. In order to meet the latency constraints for the conference, at most $\lceil 60 \times 0.150 \rceil$ = 9 frames may be packaged together and the conferencing application must generate a message at least every 9/60 = 0.15 seconds, which means the audio stream must transmit at least 7 messages per second. The area labeled Latency Constraint in Figure 3-19 shows the audio operating point (6, 120k) eliminated from consideration due to the network latency at time *t*.

The operating points eliminated by latency constraints vary over time. Suppose the network latency increases so $L_a(t)$ = 190 *ms*, which implies $Buf_a^{max}(t)$ = 60 *ms*. Now, at most $\lceil 60 \times 0.060 \rceil$ = 4 frames may be packaged together, so the message rates below 15 messages/second may not be considered. The set of operating points eliminated when the network latency is 190 *ms* is shown in Figure 3-20 in the area

labeled *Latency Constraint*. The area associated with the latency constraint has increased over that in Figure 3-19 due to the increased network latency. If network latency subsequently decreases, the constraint area also decreases and may make some operating points again available for consideration.

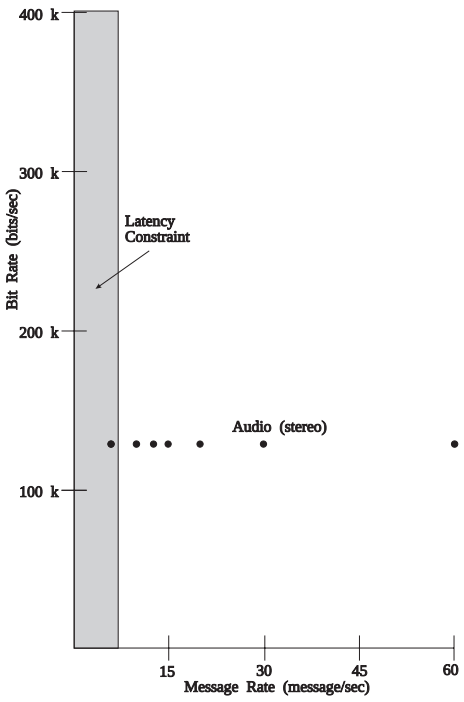


Figure 3-19: Latency Constraints for Audio when Network Latency = 100 *ms*

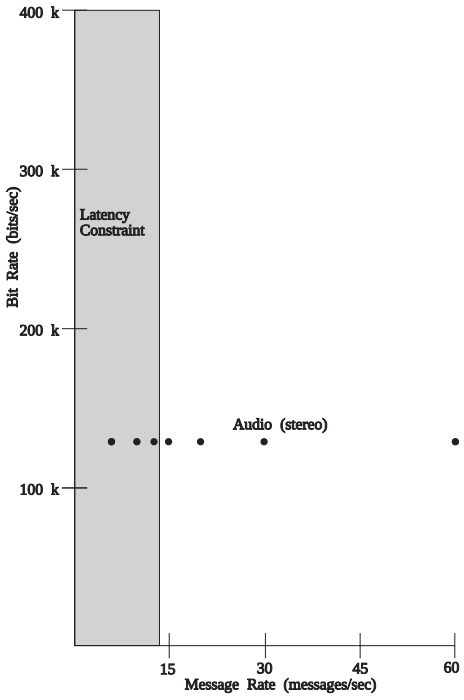


Figure 3-20: Latency Constraints for Audio when Network Latency = 190 *ms*

Note that even if the network latency were zero ($L_S(t)$ = 0 *ms*), the induced latency is still limited (*i.e.*, $Buf_s^{max}(t) = L_s^{max}$ = 250 *ms*) and some operating points may be eliminated from consideration. In the audio example, even with zero network latency, at most $\lceil 60 \times 0.250 \rceil$ = 15 frames may be packaged together which implies a minimum packet rate of 4 packets per second. Even if the conferencing system were able to package all 60 audio frames into a single packet (*i.e.*, there is an operating point at (1, 120k)), the operating point is never a candidate since the induced latency required to package all 60 frames together exceeds the maximum allowable transmission latency.

We can combine the latency and fidelity constraints to describe the set of operating points that satisfy the perceptual constraints on the system. Let

$$POP_s(t) = \left\{(m_s, b_s) | (m_s, b_s) \in OP_s, 1/m_s \le Buf_s^{max}(t) \wedge b_s \ge F_s^{min}\right\} \quad (3\text{-}1)$$

be the set of *perceptual operating points*. Points in $POP_s(t)$ satisfy minimal perceptual constraints. Operating points not in $POP_s(t)$ cannot be used at time t since they will inherently lead to either unacceptable latency or fidelity in stream s. Figure 3-21 illustrates a sample set of perceptual operating points for the conferencing system originally described with the operating points in Figure 3-4. The perceptual operating points for both audio and video are identified in Figure 3-21 by the operating points in the area labeled *Perceptual Operating Points*. In this example, audio and video have the same latency and fidelity limits ($L_a^{max} = L_v^{max} = 250$ *ms* and $F_a^{min} = F_v^{min} = 100$ kbits/second). Network latency for both streams is 190 *ms* ($L_a(t) = L_v(t) = 190$ *ms*).

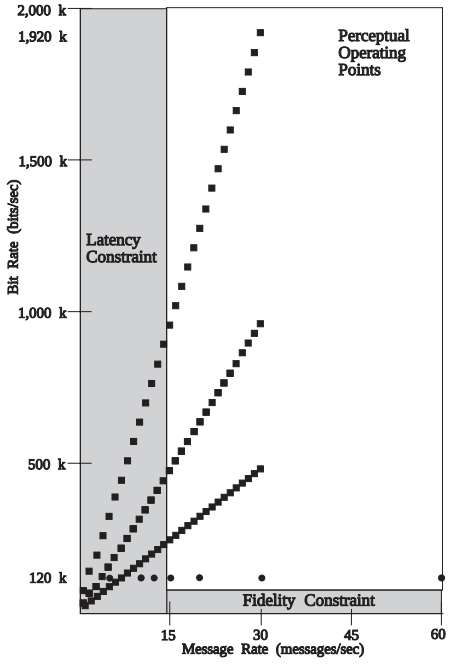


Figure 3-21: Exclusion of Operating Points due to Perceptual Constraints

3.3. Transmission Constraints

The first requirement for an operating point to be feasible is that it provides adequate perceptual quality (*i.e.*, is a member of $POP_s(t)$). The second requirement for feasibility is that the operating point (m_s, b_s) is sustainable under the current network conditions. In other words, the network must be able to successfully transmit a stream producing m_s messages/second with an aggregate bit rate of b_s bits/second. We call the constraints imposed by the network on the selection of the media operating point *transmission constraints.* The network limits our choice of an operating point in two

ways. First, for stream *s* operating point (m_s, b_s), in the absence of congestion, it may be the case that there exists a network element that is unable to process the media streams due to limitations of the physical hardware or software components of the element. For example, suppose the operating point is (30, 1920k), but the path contains a T1 line with a maximum bit rate of 1.544 megabits/second. We call such constraints *structural constraints*. Structural constraints are fundamental network limitations regardless of the traffic load in the network. Second, in the presence of congestion, there may exist a network element that is temporarily unable to sustain a b_s bit/second stream that is partitioned into $p_{s,k}$ packets/second (*i.e.,* the packet rate generated on the k^{th} hop) because of the aggregate demand for resources at the network element. For example, suppose a router can process 1,000 packets per second and that other, non-conference traffic is currently sending 800 packets per second through the router. The video conference cannot send more than 200 packets per second through the router until the non-conference traffic decreases. We call these constraints *congestion constraints*. Congestion constraints are temporary network limitations due to aggregate demand at some network bottleneck.

The primary effect of transmission constraints at a network element is the development of a queue of waiting packets. Expanding queues increase waiting times and eventually may cause packet loss. Reducing either the bit rate, b_s, or the message rate, m_s, of one or more of the conference streams may reduce the load sufficiently such that the congested network element is able to eventually empty its queue of pending requests. The characteristics of the congested element and the nature of the competing traffic determine whether reducing the bit rate or the message rate will have the most effect on the congested network element. For video conferencing, reducing the conference bit rate improves the quality of the delivered conference under certain network conditions while reducing the message rate does not and vice versa. Sometimes both work and sometimes nothing works.

3.3.1. A Simple Network Queueing Model

This section introduces a simple queueing model to describe the effects of message and bit rates on performance and the relationships of these rates to network congestion. The models are taken from a branch of general queueing theory called the operational analysis of queueing network models [28, 68] that is widely used to model computer systems. Operational queueing models are good vehicles for introducing the

intuition behind why the transmission control framework works. Since the transmission control framework is based on end-to-end adaptations over existing networks, it is difficult or impossible for the conference endpoints to collect all the data (*e.g.*, network topology, network technology, traffic load, *etc.*) required to evaluate the actual state of the network via a queueing network. Given this fact, the purpose of this section is not to build and evaluate a complete queueing model for a particular network. The goal is to use some simple queueing models to explain the problems associated with transmission of continuous media in a constrained network and describe how the transmission control framework addresses these problems.

Recall the definitions of network nodes, hops, and paths discussed earlier (see Figure 3-10). A single hop is modeled as two servers and their associated queues (Figure 3-22).

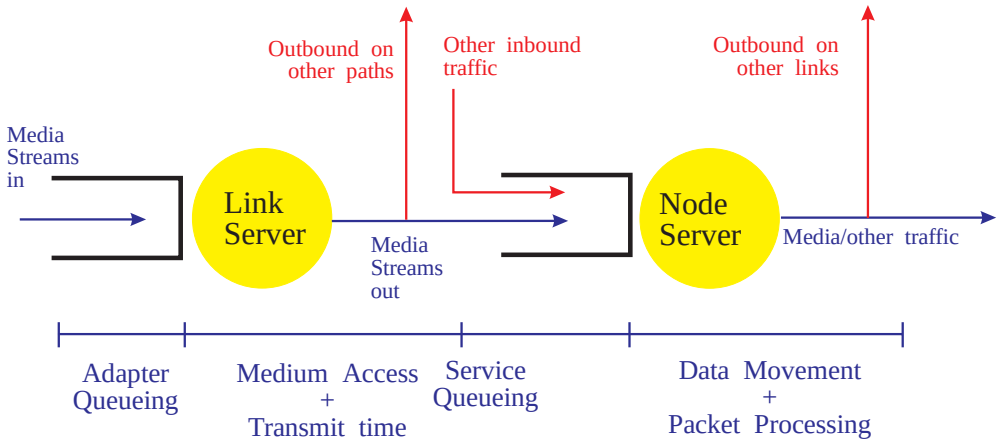


Figure 3-22: Simple Model of a Network Hop

The first server in Figure 3-22 represents a network adapter in a node and the associated outbound network link. The second server represents the next node in the network path. We call the first server the *link server* and the second the *node server*. Figure 3-23 shows how the link and node servers span two computers and the link connecting those computers. Link servers correspond to the *link n* elements in Figure 3-10 and node servers correspond to the *node n* elements. The network path is composed of *h hops* and is modeled as a series of hops (*i.e.*, a series of instances of Figure 3-22). The link server represents acquiring control of the transmission medium

and actual transmission on the medium. The node server represents the packet processing and data movement time within a node.

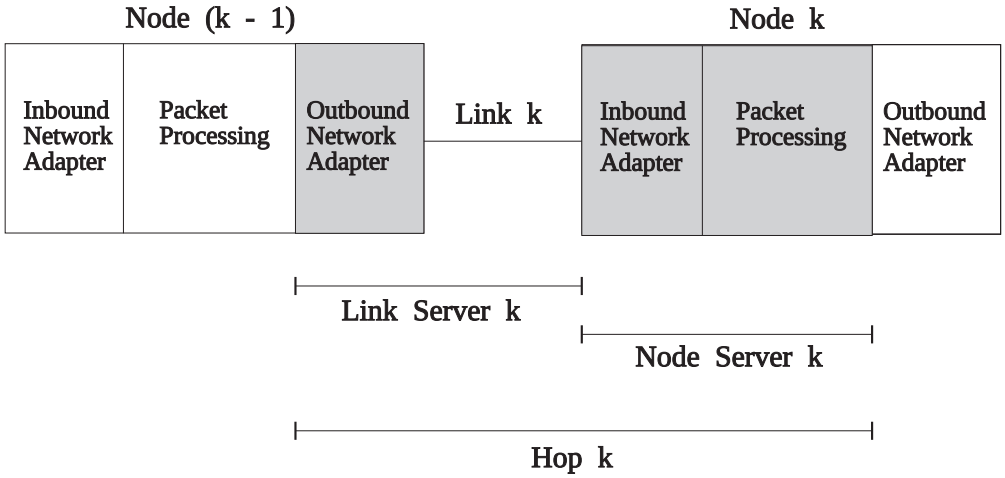


Figure 3-23: Mapping Physical Components to Logical Servers

For a video conference, there are typically three classes of customers at a server: an audio class *a*, a video class *v*, and the non-conference class *d*. Messages correspond to jobs and there may be many jobs in each class. *C* is the set of all customer classes (*i.e.*, $C = \{a, v, d\}$). The service demand for a particular class $c \in C$ is $D_{c,k}^{link}$ and $D_{c,k}^{node}$ for the link and node servers, respectively. A server is *saturated* if the utilization of the server reaches 1, so if each class *c* has an associated message rate m_c, the link or node server at hop *k* are saturated if the corresponding relation below holds.

$$1 \leq \sum_{c \in C} m_c D_{c,k}^{link}$$

$$1 \leq \sum_{c \in C} m_c D_{c,k}^{node}$$

In theory, if a server is saturated, the queue of jobs (*i.e.*, messages) waiting at the server continually grows and jobs are delayed for an arbitrarily long time [68]. In the physical network, a job is represented by packets in router work queues. If the servers become saturated, eventually the queue fills all available buffers and packets are delayed or lost. Both of these situations adversely affect the quality of the conference, so we want to avoid saturating any server along the path.

We are interested in how a single media stream can adapt to network conditions even when other data streams (*e.g.*, jobs in class *d*) do not adapt, so from this point we will consider server utilization and saturation from the perspective of a single media stream $s \in \{a, v\}$. It is perfectly valid to view adaptation using the aggregate characteristics of the entire video conference (*i.e.,* consider the *a* and *v* streams together), but for simplicity we focus here on independent control of the streams. It will prove useful to refer to the utilization of a server resulting from all work done for streams other than *s*. $U_{other,k}^{link}$ is this utilization for the link server and $U_{other,k}^{node}$ is this utilization for the node server. The definitions of $U_{other,k}^{link}$ and $U_{other,k}^{node}$ are:

$$U_{other,k}^{link} = \sum_{c \in C, c \neq s} m_c D_{c,k}^{link}$$

$$U_{other,k}^{node} = \sum_{c \in C, c \neq s} m_c D_{c,k}^{node}$$

The stream *s* saturates the link or node server on hop *k* if the relations below are true.

$$1 - U_{other,k}^{link} \leq m_s D_{s,k}^{link}$$

$$1 - U_{other,k}^{node} \leq m_s D_{s,k}^{node}$$

In the queueing model, a job corresponds to a message from a particular class. For a message in class $s \in \{a, v\}$, the service demand for a class at a server is the service time required for a single visit to the server, $S_{s,k}^{link}$ or $S_{s,k}^{node}$, multiplied by the number of times the message visits the server, $V_{s,k}$. A visit corresponds to the processing of a single packet that is a component of a perhaps larger message. When transmitting messages along the network path, the number of times a message visits hop *k* corresponds to the number of packets generated on hop *k* as a result of message rate m_s. For a given hop $k \in \{1..h\}$, the visit count for a particular class, $V_{s,k}$, is the same for both the link and node servers, so the demand per message at each server type is given by the equations below.

$$D_{s,k}^{link} = S_{s,k}^{link} V_{s,k}$$

$$D_{s,k}^{node} = S_{s,k}^{node} V_{s,k}$$

We can calculate the number of visits for a given hop if we know the relationship between packets and messages. In section 3.1.3, we discussed how we can calculate

the packet rate for stream s on hop k, $p_{s,k}$, from the operating point (m_s, b_s). Given this calculation, we can calculate the visit count for a particular class $s \in \{a, v\}$ for a particular hop $k \in \{1..h\}$ using the equation below.

$$V_{s,k} = \frac{p_{s,k}}{m_s}$$

We must determine the service time per visit differently for link servers and node servers. For link servers, the service time per visit is the time it takes to acquire exclusive control of the transmission link plus the time it takes to transmit the packet over the link. Let MA_k be the average *medium access time* or the time it takes to gain control the transmission link on hop k. On a token ring network, MA_k corresponds to the average time needed to acquire a free token. For an Ethernet network, MA_k corresponds to the average time it takes to acquire an idle carrier. MA_k varies over time and may be different if measured over different intervals. The time required to transmit a packet depends upon the speed of the network link and the size of the packet. Let r_k^{link} be the transmission rate of the link on hop k. Since the server acquires exclusive access to the transmission link, the transmission rate is constant for all packets and does not vary over time. We calculate the average size of the packet for a given operating point (m_s, b_s) by dividing the stream bit rate by the packet rate on hop k, which is $b_s/p_{s,k}$. The service time per visit for stream s on the link server at hop k is:

$$S_{s,k}^{link} = MA_k + \frac{b_s}{p_{s,k} r_k^{link}}$$

Using the service time per visit and the visit count, we can now describe the service demand per message at the link server on hop $k \in \{1..h\}$ for stream $s \in \{a, v\}$ as follows.

$$D_{s,k}^{link} = S_{s,k}^{link} V_{s,k} = \left(MA_k + \frac{b_s}{p_{s,k} r_k^{link}} \right) V_{s,k} = \frac{MA_k \, p_{s,k}}{m_s} + \frac{b_s}{m_s r_k^{link}}$$

For node servers, the service time per visit is the time it takes to process a packet (*e.g.*, examine the packet and determine the outbound route) plus the data transfer time within the node. Let PP_k be the average *packet processing time*. Data movement time within the node depends upon the speed of the internal data transfer

and the size of the packet. Let r_k^{node} be the internal data transfer rate of the node on hop k. This rate is a function of the node hardware and software (*e.g.*, CPU, bus, *etc.*). As with the link server, we calculate the size of the packet by dividing the stream bit rate by the packet rate on hop k, which is $b_s/p_{s,k}$. We calculate the service time per visit on the node server at hop k as follows.

$$S_{s,k}^{node} = PP_k + \frac{b_s}{p_{s,k} r_k^{node}}$$

Using the service time per visit and the visit count, we can now describe the service demand per message at the node server on hop $k \in \{1..h\}$ for stream $s \in \{a, v\}$ as follows.

$$D_{s,k}^{node} = S_{s,k}^{node} V_{s,k} = \left(PP_k + \frac{b_s}{p_{s,k} r_k^{node}} \right) V_{s,k} = \frac{PP_k \, p_{s,k}}{m_s} + \frac{b_s}{m_s r_k^{node}}$$

The link server on hop k is *constrained* for $(m_s, b_s) \in OP_s$ if the following relation holds.

$$1 - U_{other,k}^{link} \le m_s D_{s,k}^{link}$$

$$1 - U_{other,k}^{link} \le m_s \left(\frac{MA_k \, p_{s,k}}{m_s} + \frac{b_s}{m_s r_k^{link}} \right)$$

$$1 - U_{other,k}^{link} \le MA_k p_{s,k} + \frac{b_s}{r_k^{link}} \qquad (3\text{-}2)$$

The right hand side of relation (3-2) is the link server utilization required to acquire control of the medium and transmit the data for all packets associated with class s across the link. When the link server is constrained, this utilization exceeds the available utilization (*i.e.*, the utilization unused by all other classes of traffic besides s).

The node server on hop k is *constrained* for $(m_s, b_s) \in OP_s$ if the following relation holds.

$$1 - U_{other,k}^{node} \le m_s D_{s,k}^{node}$$

$$1 - U_{other,k}^{node} \le m_s \left(\frac{PP_k \, p_{s,k}}{m_s} + \frac{b_s}{m_s r_k^{node}} \right)$$

$$1 - U_{other,k}^{node} \le PP_k p_{s,k} + \frac{b_s}{r_k^{node}} \qquad (3\text{-}3)$$

The right hand side of relation (3-3) is the node server utilization required to process and forward all packets associated with class *s* through the node. When the node server is constrained, this utilization exceeds the available utilization.

When the link server or the node server is constrained on any hop, operating point (m_s, b_s) is not sustainable because some server in the path is saturated. The type of constraint present at the link or node server depends upon whether the constraint is primarily due to the offered bit or message load. The following sections discuss the differences between these two types of constraints and the implications of these differences.

3.3.2. Capacity Constraints

3.3.2.1. Definitions of Capacity Constraints

A *structural capacity constraint* exists for (m_s, b_s) $\in$ OP_s if the following holds for any hop *k* in the path from source to sink.

$$\left(1 \le \frac{b_s}{r_k^{link}}\right) \vee \left(1 \le \frac{b_s}{r_k^{node}}\right)$$

In other words, a structural capacity constraint exists for (m_s, b_s) $\in$ OP_s if there is a hop *k* where either the link transmission rate or the internal data transfer rate at the node is too slow for the offered bit rate. The offered bit rate saturates a server on hop *k* in the path even with no competing work from other classes (*i.e.*, $U_{other,k}^{link} = U_{other,k}^{node} = 0$) or any service time due to media access or packet processing (*i.e.*, $MA_k p_{s,k} = PP_k p_{s,k} = 0$).

Structural capacity constraints exist because of physical data transfer limits in the network path hardware or software. Physical network changes are required to relieve structural capacity constraints. As such, operating points that are structurally capacity constrained are inherently infeasible and should never be considered.

Consider the video conferencing system with the operating points shown in Figure 3-4 and the network in Figure 3-24. For this particular video conferencing system, the

network has a structural capacity constraint since the transmission rate of the T1 link (1.544 megabits/second) is insufficient to carry the full video stream (*i.e.*, the highest video bit rate is 1.92 megabits/second). The area labeled *Structural Capacity Constraint* in Figure 3-25 graphically represents this constraint. The structural constraint only affects the video stream (*i.e.*, only the video stream has operating points in the area labeled *Structural Capacity Constraint*). The low bit rates associated with audio are not constrained. The video stream is much more likely to experience the effects of structural capacity constraints than audio because of the high bit rate of video relative to audio. Fortunately, local networks rarely experience structural capacity constraints due to the relatively high bit rates of local area networks. On the other hand, when conference streams cross wide-area networks, structural capacity constraints are much more common because of the relatively low bit rates of wide-area telecommunications links.

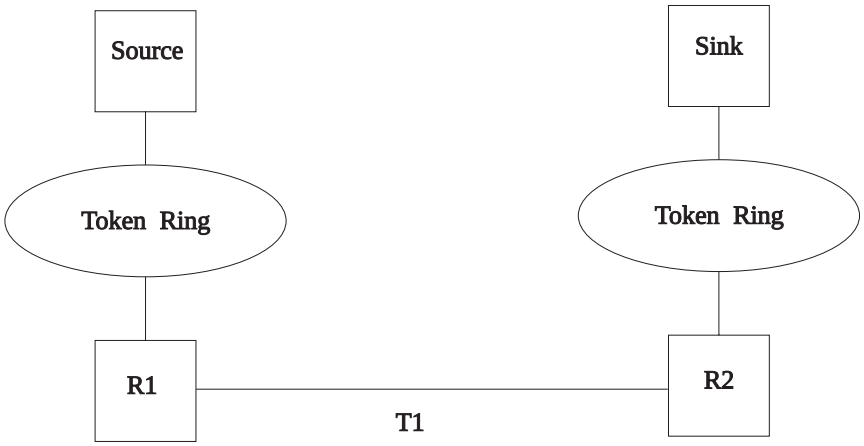


Figure 3-24: Sample Structurally Constrained Network

A *congestion capacity constraint* exists for $(m_S, b_S) \in OP_S$ if the following holds for any hop k in the path from source to sink.

$$\left(1 - U_{other,k}^{link} \leq \frac{b_s}{r_k^{link}}\right) \vee \left(1 - U_{other,k}^{node} \leq \frac{b_s}{r_k^{node}}\right) \qquad (3\text{-}4)$$

Congestion capacity constraints result when the demand for service at a server is such that the offered bit rate from stream s cannot be sustained. In the absence of other work at the server, the bit rate may be sustainable (*i.e.*, there is no structural capacity constraint), but under the current workload, the server is saturated. The constraint

exists because of the offered bit rate and exists regardless of the offered message or packet rate.

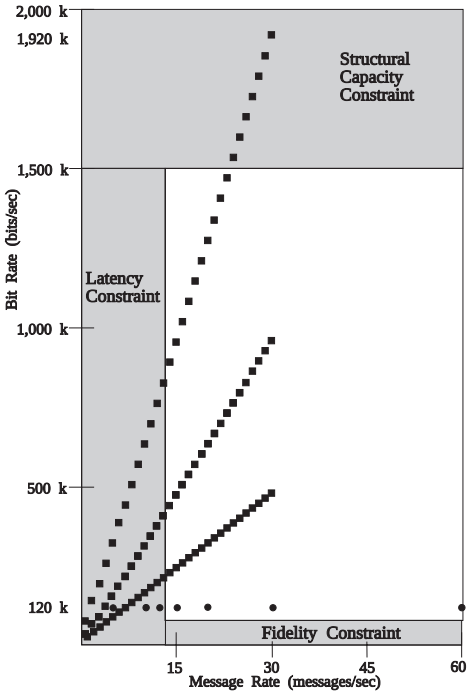


Figure 3-25: Exclusion of Operating Points due to Structural Capacity Constraints

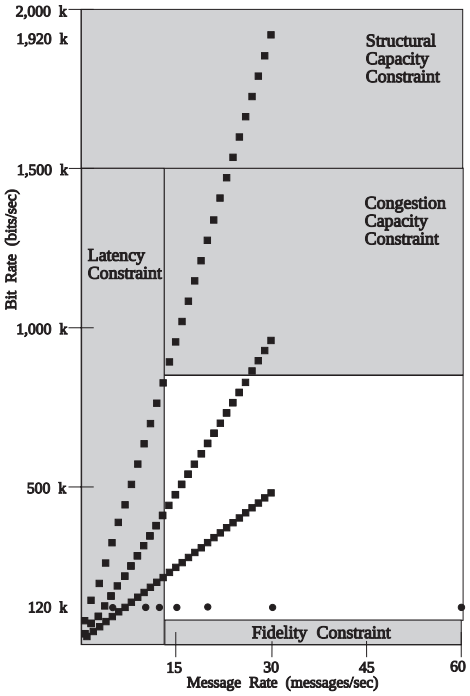


Figure 3-26: Congestion Capacity Constraint

Since the transmission control algorithm for stream *s* cannot directly affect $U^{link}_{other,k}$ or $U^{node}_{other,k}$, the only way to address the constraint is for the conference to select an operating point for *s* with a lower bit rate. Varying the message rate without changing the bit rate will not remove the constraint since the service required to support the offered bit rate is too large regardless of the message rate. Since $U^{link}_{other,k}$ and $U^{node}_{other,k}$ may vary when measured over different intervals, a given operating point may be congestion capacity constrained during some intervals and not during others. Figure 3-26 gives a graphical representation of a congestion capacity constraint. All operating points within the area labeled *Congestion Capacity Constraint* have congestion capacity constraints and cannot be successfully used by the conference. We say $(m_s, b_s) \in OP_s$ is *capacity constrained* if it has a structural or congestion capacity constraint.

Stream *s* is *fundamentally capacity constrained* if the following holds for any hop *k* in the path from source to sink.

$$\forall (m_s, b_s) \in OP_s, \left(1 - U_{other,k}^{link} \leq \frac{b_s}{r_k^{link}}\right) \vee \left(1 - U_{other,k}^{node} \leq \frac{b_s}{r_k^{node}}\right)$$

If *s* is fundamentally capacity constrained, given the current environment, the transmission strategy cannot address the capacity constraint by manipulating the single stream *s* since no operating point can be successful. It is possible that the conference may still be able to relieve the constraint by manipulating multiple media streams in concert, but as mentioned earlier, this work focuses on independent stream control and we do not explicitly consider direct coordinated changes to multiple streams (there is indirect coordination due to the interaction caused by independent reactions to the congestion by each stream). If desired, a conferencing system could use an aggregate stream of all media streams to provide a single coordinated response across all media streams.

3.3.2.2. Experimental Demonstration of Capacity Constraint Effects

We can use the network shown in Figure 3-27 to illustrate the effects of a capacity constraint on a video conference. The conferencing system in these experiments has a default video operating point of (30, 1920k) and a default audio operating point of (60, 120k). The token rings in the network in Figure 3-27 are 16 megabit/second rings. The size of the payloads on the token rings is about 17,800 bytes. Since the maximum payload is larger than any audio or video frame and a message only carries one frame, the media message rates equal their frame rates (*i.e.*, $m_S = p_{s,k}$ for all hops *k*). Normally, routers *R1* and *R2* are able to carry the full aggregate conference traffic, so there is no structural capacity constraint. To generate a capacity constraint, the router *R1* runs a modified routing program that limits the internal transfer rate of the router to a specified level. In our first experiment, the routing software limits the transfer rate through the router to approximately 1.5 megabits/second (*i.e.*, $r_1^{node} = 1.5$ megabits/second -- approximately the speed of a T1 transmission link). The service demand for the video stream creates a bottleneck at router *R1*. In particular, the node server associated with *R1* has a congestion constraint resulting from data movement within the node since $\frac{b_v}{r_1^{node}} = \frac{1.920}{1.5} = 1.28 > 1.$

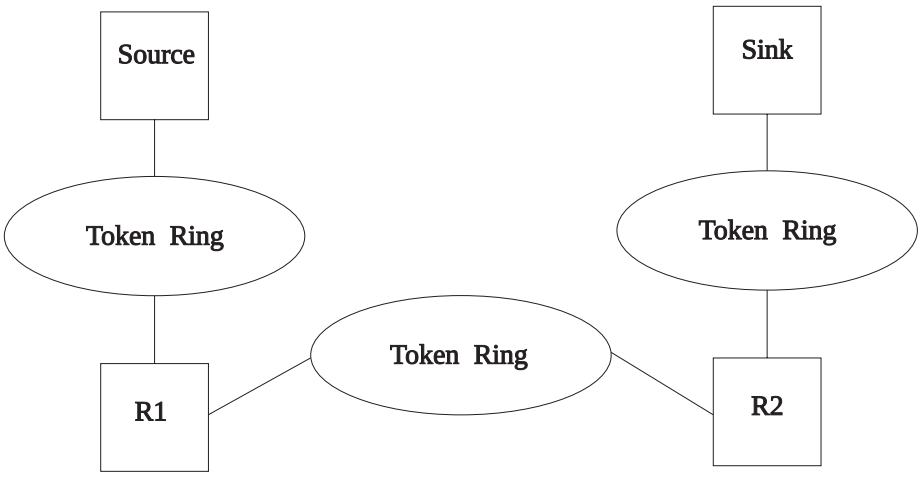
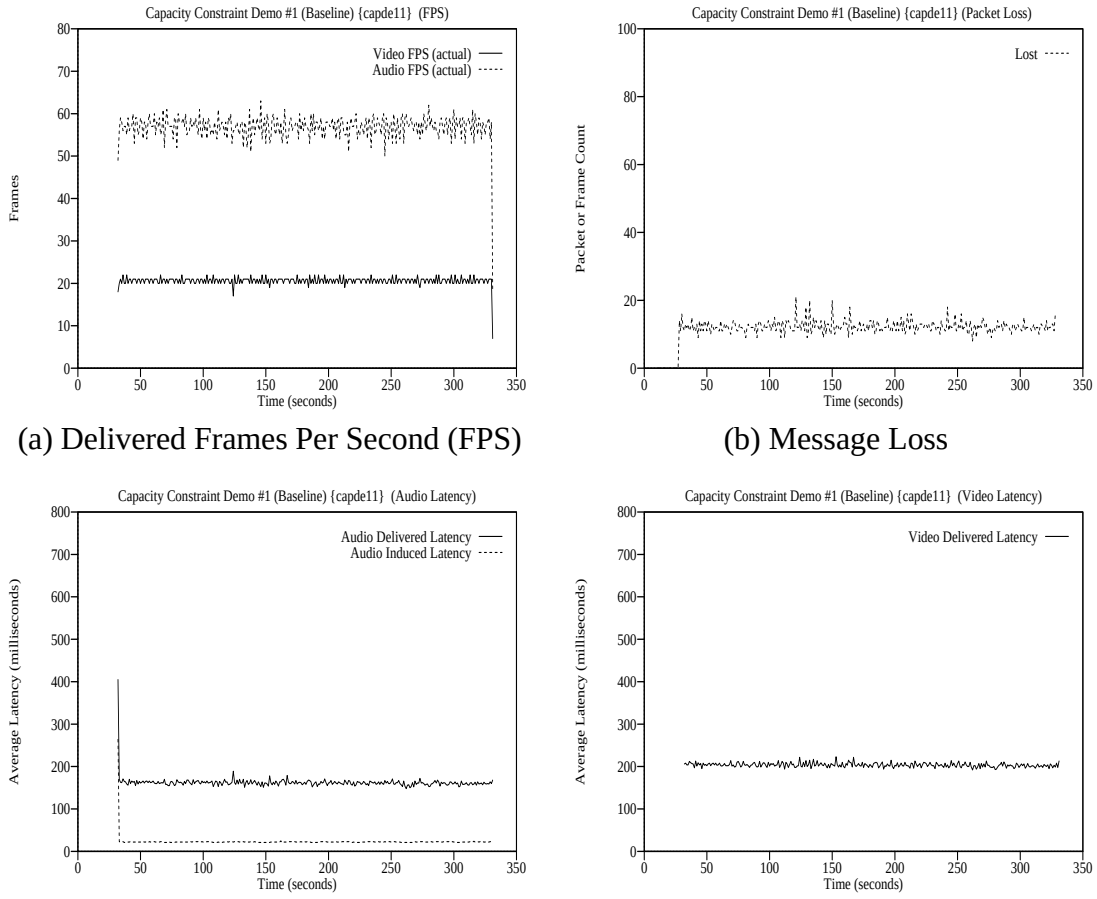


Figure 3-27: Experimental Network Configuration

Figure 3-28 shows the results when the conferencing application uses the default operating points for both the audio and video streams. The application transmits video using the high quality encoding and at 30 frames per second. The application transmits audio at 60 frames per second and in stereo. We refer to this conference as the *Baseline* case. Figure 3-28 (a) shows the number of audio and video frames delivered at the conference destination in each second of the conference. The x-axis of the graph is the seconds into the video conference. The y-axis is the number of frames delivered during the second. Video frame delivery is shown as a solid line and audio delivery as a dashed line. Figure 3-28 (b) shows the number of messages lost during each second of the conference. The x-axis is seconds into the conference and the y-axis is the number of messages lost. The number of messages includes both audio and video messages. Figure 3-28 (c) and (d) show the stream latency for audio and video, respectively. The x-axis is the number of seconds into the conference and the y-axis is the latency in milliseconds.



(a) Delivered Frames Per Second (FPS)

(b) Message Loss

(c) Audio Latency

(d) Video Latency

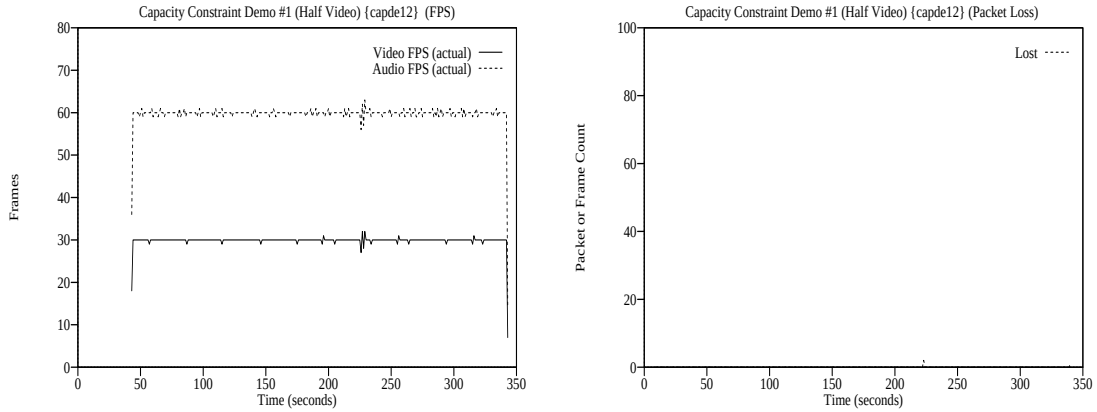
Figure 3-28: Capacity Constraint #1 - *R1* limited to 1.5 Mb/s (Figure 3-27) - Baseline

Figure 3-28 (a) shows the delivered frame rate for the audio stream varies between about 50 and 60 frames per second. Unfortunately, audio delivery is inconsistent and there are over 900 audio gaps (Table 3-1), resulting in poor perceived audio quality. Video is delivered at about 20 frames per second. Part (b) of the figure shows that there are frames lost in transit throughout the duration of the conference. Since the service demand for video exceeds the maximum the router can support, the video stream eventually fills the available buffers in *R1* and many packets are lost due to buffer overflow. Those packets that are not dropped still experience a significant queueing delay, as shown by the stream latency graphs in parts (c) and (d) of Figure 3-28.

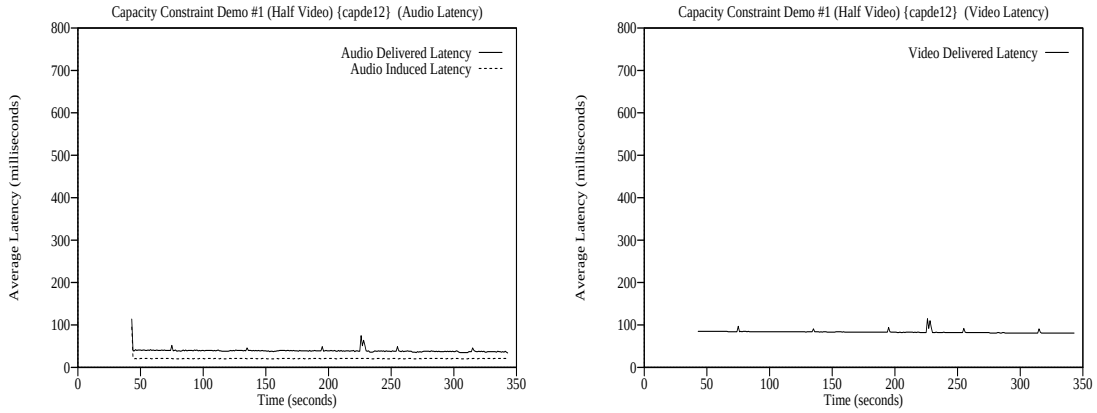
Figure 3-29 shows the results obtained with a conference using a new video operating point (30, 960k) with the same audio operating point (60, 120k). We call this the *Medium Quality Video* case. The change in video operating points comes from using

a medium quality video encoding that generates approximately half the bit rate of the high quality encoding used in the previous experiment. Here the congestion capacity constraint experienced by the video stream has been relieved, since $\frac{b_v}{r_1^{node}} = \frac{.960}{1.5} = .64 < 1.$

Figure 3-29 shows the improvement in the conference. Audio and video frame rates are delivered at the transmitted frame rates with less variability (part (a)) and with no message loss (part (b)). The number of audio gaps drops to 20 and the video frame rate is about 30 frames per second (Table 3-1). Audio and video latencies are over 100 *ms* lower than those in the *Baseline* case (compare parts (c) and (d) in Figures 3-28 and 3-29 and the summaries in Table 3-1). Of course, this improvement comes at the expense of video quality, but subjectively the improvements in frame rates and latencies offset the reduction in image quality. The audio is perfect and the high video frame rates limit the impact of the lower video quality, particularly for the casual observer. Furthermore, the conference is a better network citizen. Its transmission rate is sustainable in the network (the number of messages transmitted is the number received). The operating points used in the *Medium Quality Video* case completely ameliorate the effects of the capacity constraint.



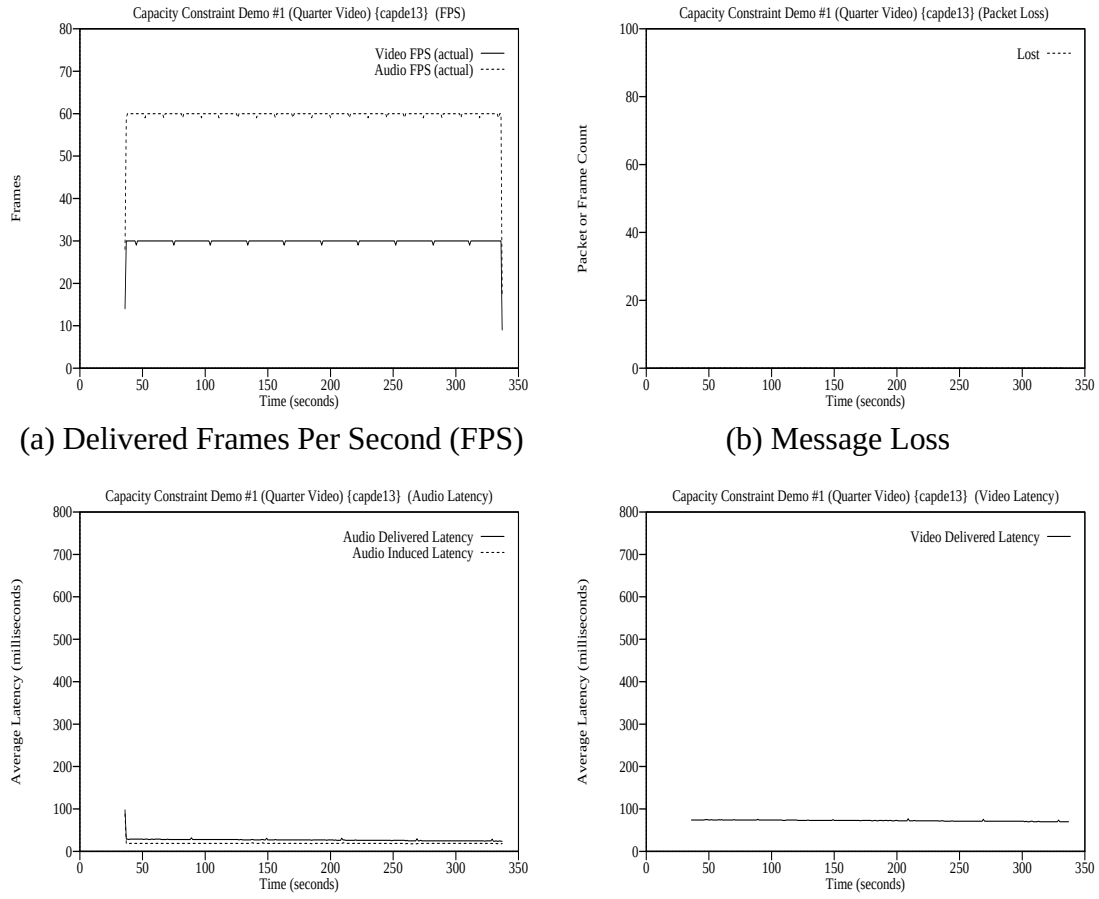
(a) Delivered Frames Per Second (FPS) (b) Message Loss



(c) Audio Latency (d) Video Latency

Figure 3-29: Capacity Constraint #1 - *R1* limited to 1.5 Mb/s (Figure 3-27) - Medium Quality Video

Figure 3-30 shows the results for the *Low Quality Video* case. This conference uses a low quality video encoding corresponding to the video operating point (30, 480k). The delivered conference quality is lower than in the *Medium Quality Video* case. Frame throughput and latency are essentially identical to the *Medium Quality Video* case, but the video fidelity is lower because of the lower quality video encoding. The point here is that once the capacity constraint has been relieved, further reductions in bit rate are not needed and unnecessarily reduce the conference fidelity.



(a) Delivered Frames Per Second (FPS)

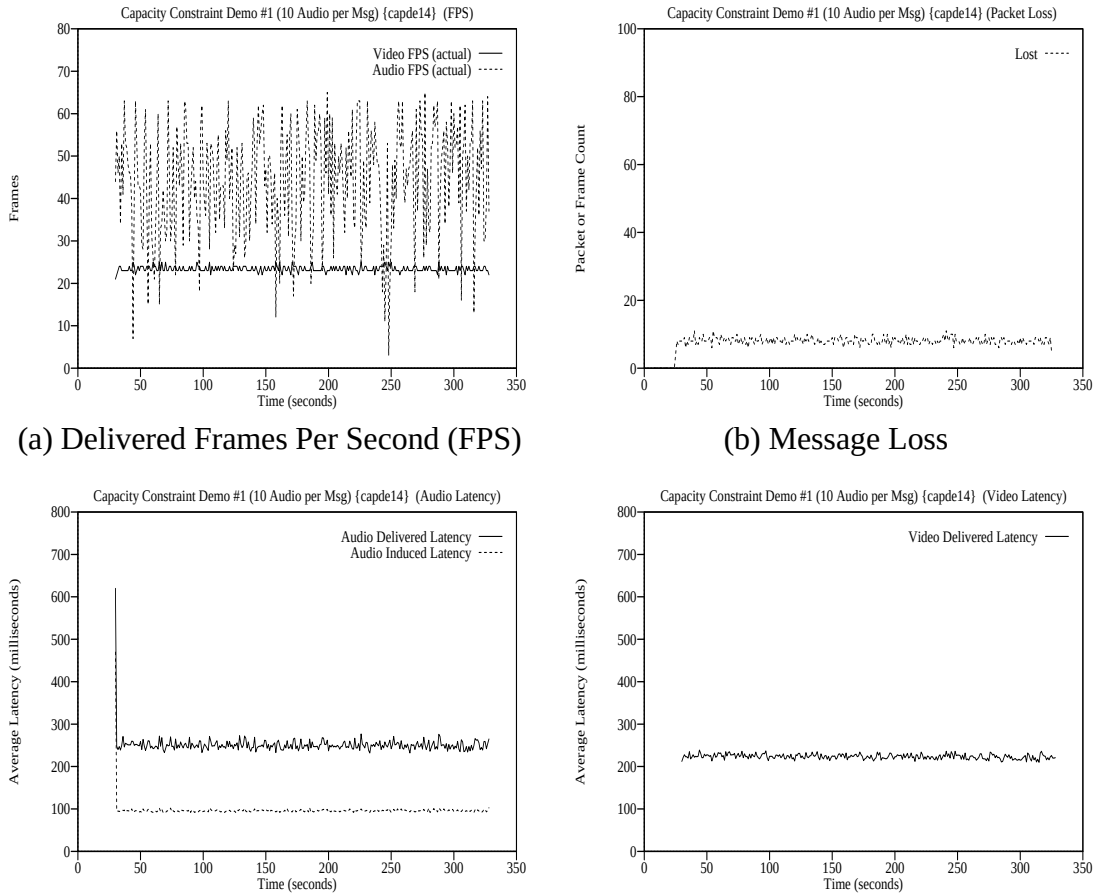
(b) Message Loss

(c) Audio Latency

(d) Video Latency

Figure 3-30: Capacity Constraint #1 - *R1* limited to 1.5 Mb/s (Figure 3-27) - Low Quality Video

In Figure 3-31, rather than lowering the video bit rate to address the congestion capacity constraint, we instead reduce the audio message rate to 6 messages/second. In this case, each audio message carries 10 audio frames. The video operating point is (30, 1920k) and the audio operating point is (6, 120k). Audio frame delivery (part (a)) is worse than in any previous case. There are over 4600 audio gaps in this case compared with about 900 in the *Baseline* case and about 20 with the *Medium Quality Video* and *Low Quality Video* cases (see Table 3-1). The large number of gaps is due to the high message loss (part (b)). Audio fidelity suffers significantly when an audio message is lost since each message carries 10 audio frames. Since the change in message rate does not reduce the number of bits that *R1* must process (*i.e.*, the aggregate bit rate is the same as the first experiment), the change in audio operating points does not remove the capacity constraint at *R1* and conference quality actually deteriorates.



(a) Delivered Frames Per Second (FPS)

(b) Message Loss

(c) Audio Latency

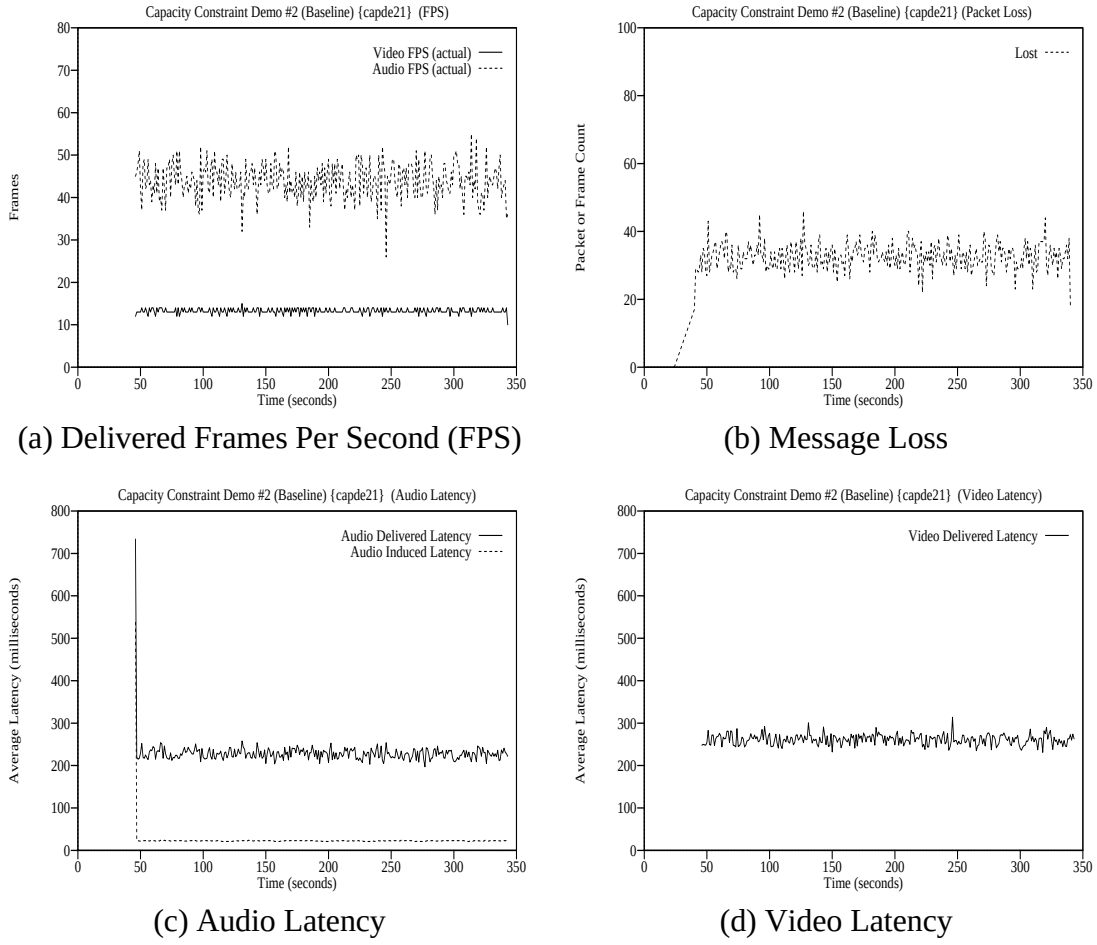
(d) Video Latency

Figure 3-31: Capacity Constraint #1 - *R1* limited to 1.5 Mb/s (Figure 3-27) - Audio 10 Frames/Msg

Capacity demo #1	Baseline	Medium Video	Low Video	10 Audio/Msg
Audio FPS				
Mean	56.85	59.78	59.79	44.42
Standard deviation	3.15	2.72	2.49	12.80
Minimum	18	14	17	3
Maximum	63	63	60	65
Mode/Median	58/57	60/60	60/60	53/46
Gaps	917	20	19	4653
Audio Latency: Delivered (Induced)				
Mean	161.47 (22.16)	39.06 (20.69)	26.75 (18.97)	249.98 (95.66)
Standard deviation	4.81 (0.75)	3.37 (0.46)	1.43 (0.30)	8.95 (2.36)
Minimum	148 (20)	34 (20)	24 (18)	232 (92)
Maximum	189 (25)	75 (21)	32 (20)	277 (103)
Mode/Median	161/161 (22/22)	39/39 (21/21)	28/27 (19/19)	251/249 (95/95)
Intervals > 250 ms	0	0	0	126
Audio Messages				
Frames Sent	18008	18008	18086	17930
Msgs(frames) Lost	896 (896)	1 (1)	0 (0)	464 (4640)
Mean Frames Lost	2.98	0.00	0.00	15.47
Max Frames Lost	10	1	0	60
Video FPS				
Mean	20.63	29.89	29.90	23.39
Standard deviation	1.05	1.37	1.22	0.69
Minimum	7	7	9	22
Maximum	22	32	30	25
Mode/Median	21/21	30/30	30/30	23/23
Gaps	2805	12	12	1983
Video Latency: Delivered				
Mean	203.68	83.26	72.43	223.63
Standard deviation	5.16	3.16	1.46	5.78
Minimum	192	81	70	210
Maximum	223	115	77	239
Mode/Median	202/203	84/83	74/73	223/223
Intervals > 250 ms	0	0	0	0
Video Messages				
Frames Sent	9005	9005	9043	8995
Frames Lost	2796	2	0	1973
Mean Frames Lost	9.29	0.01	0.00	6.55
Max Frames Lost	13	1	0	9

Table 3-1: Summary of Capacity Constraint Demonstration #1

Figures 3-32, 3-33, 3-34, and 3-35 show the same set of cases as above, but with the router *R1* limited to only 1.0 megabits/second (see Figure 3-27). Table 3-2 summarizes the results for each of these cases. With the 1.0 megabit/second constraint, neither the high or medium quality video encodings (Figures 3-32 and 3-33, respectively) lower the video bit rate enough to eliminate the capacity constraint. The low quality video stream produces a bit rate below the constrained maximum and delivers a conference with high frame rates, low latency, and low loss. As with the first set of experiments, packing 10 audio frames per message (Figure 3-35) does not resolve the capacity constraint.



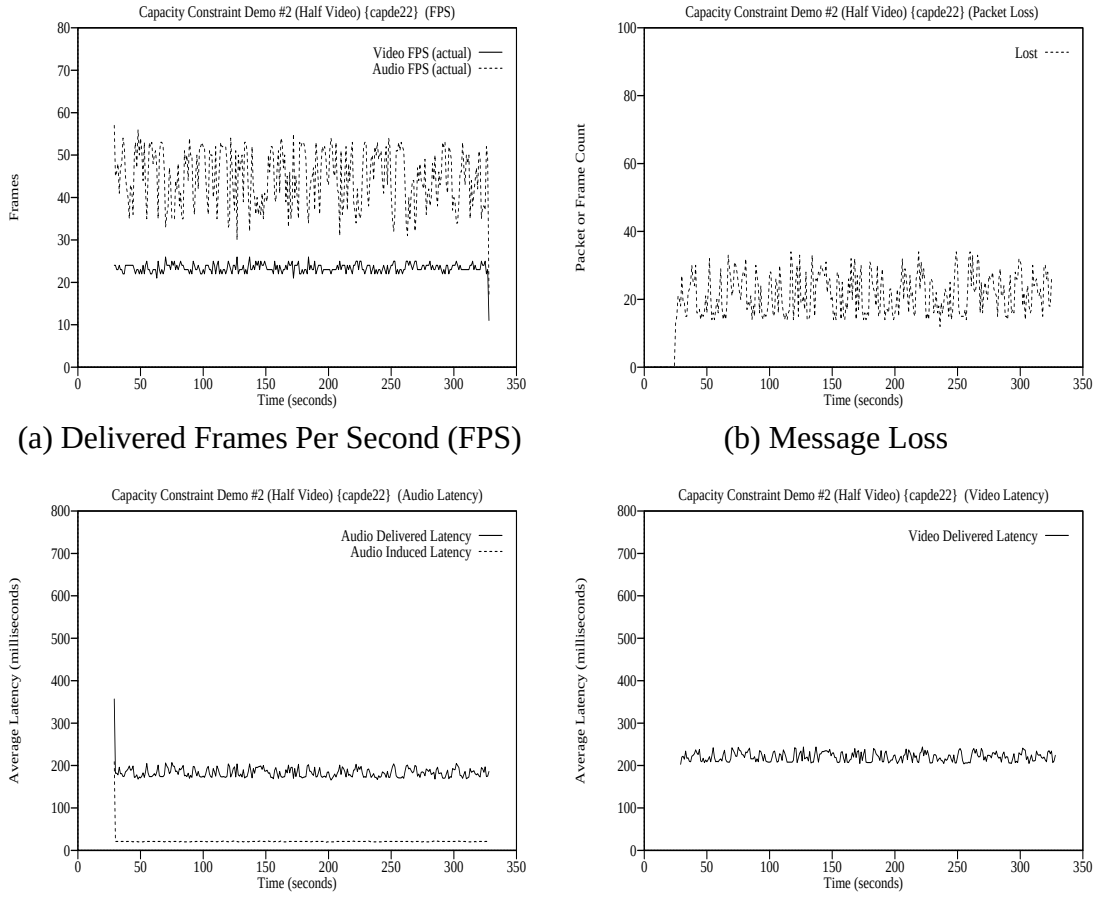
(a) Delivered Frames Per Second (FPS) (b) Message Loss

(c) Audio Latency (d) Video Latency

Figure 3-32: Capacity Constraint #2 - *R1* limited to 1.0 Mb/s (Figure 3-27) - Baseline

These demonstrations show that capacity constraints can adversely affect the quality of a conference. The key to ameliorating the effect of the capacity constraint is to reduce the offered bit rate of one or more of the conference media streams. The degree of the capacity constraint governs the degree of bit rate reduction required. Changes to

message rates alone do not address the cause of capacity constraints and are ineffective at adapting to the constraint.



(a) Delivered Frames Per Second (FPS)

(b) Message Loss

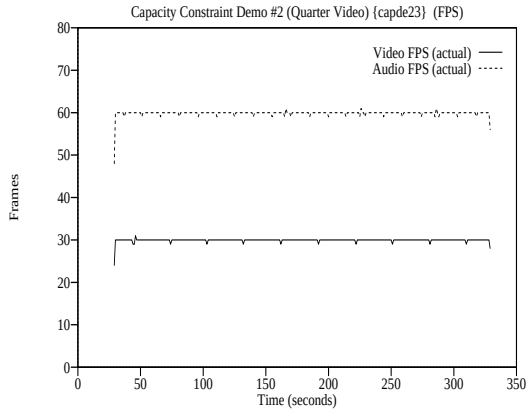
(c) Audio Latency

(d) Video Latency

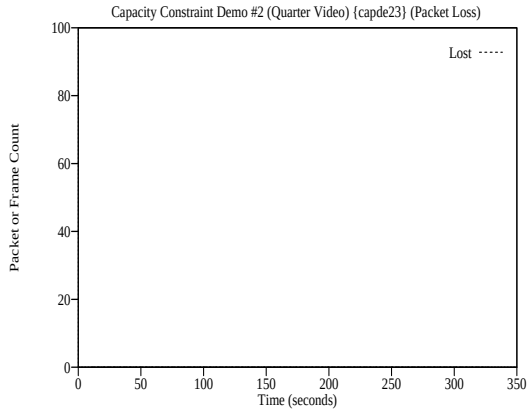
Figure 3-33: Capacity Constraint #2 - *R1* limited to 1.0 Mb/s (Figure 3-27) - Medium Quality Video

Capacity constraints are not a new concept. It is common to view network capacity as a limiting factor, particularly when dealing with wide-area telecommunications links where the link speeds are often inadequate to support the full potential bit rates of a video conference. Here, to create a capacity constraint on a typical campus LAN, we artificially constrained a router to a reduced internal data transfer time. In general, we believe existing campus LANs are not often capacity constrained due to the relatively high data transfer speeds of the LAN mediums and existing routers. We believe campus LANs and the associated routers are seldom structurally capacity constrained. Although it is possible for routers to experience congestion capacity constraints, we believe it is much more common for routers to experience congestion due to

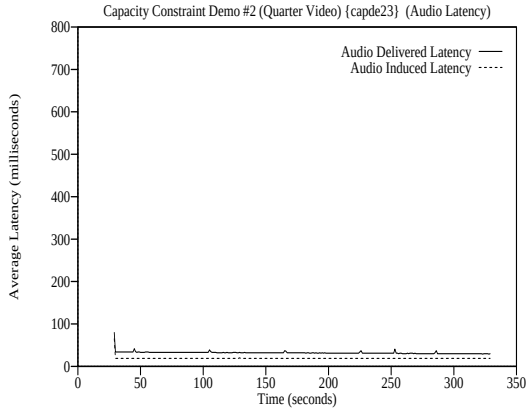
competition for control of the LAN medium. The next section discusses such constraints, which we call *access constraints.*



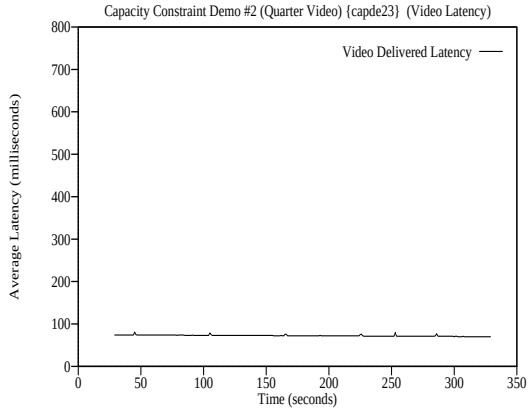
(a) Delivered Frames Per Second (FPS)



(b) Message Loss

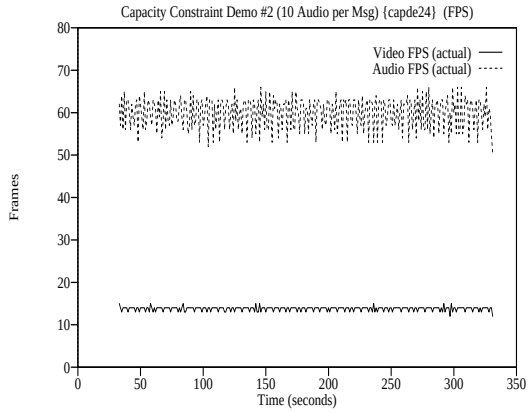


(c) Audio Latency

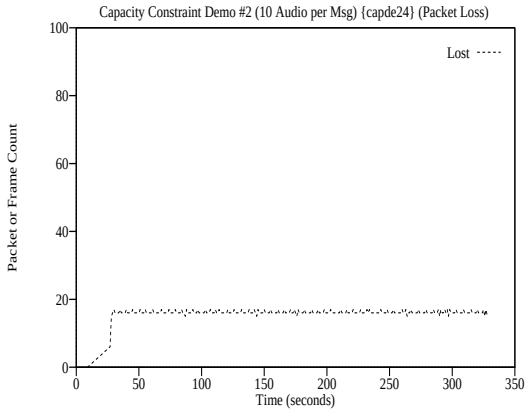


(d) Video Latency

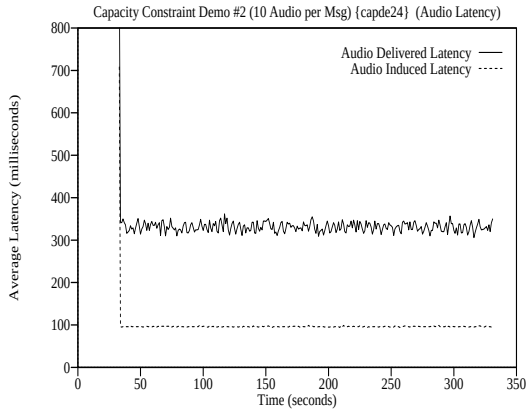
Figure 3-34: Capacity Constraint #2 - *R1* limited to 1.0 Mb/s (Figure 3-27) - Low Quality Video



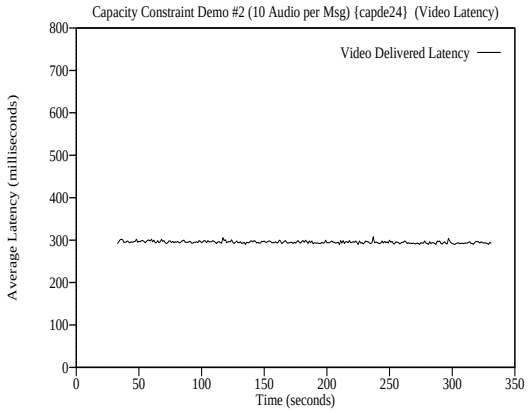
(a) Delivered Frames Per Second (FPS)



(b) Message Loss



(c) Audio Latency



(d) Video Latency

Figure 3-35: Capacity Constraint #2 - *R1* limited to 1.0 Mb/s (Figure 3-27) - Audio 10 Frames/Msg

Capacity demo #2	Baseline	Medium Video	Low Video	10 Audio/Msg
Audio FPS				
Mean	43.91	44.72	59.92	59.89
Standard deviation	4.22	6.65	0.37	3.70
Minimum	26	17	56	50
Maximum	55	56	61	66
Mode/Median	43/44	52/45	60/60	63/61
Gaps	4805	4553	18	25
Audio Latency: Delivered (Induced)				
Mean	227.45 (22.46)	182.91 (20.78)	31.85 (19)	330.95 (96.11)
Standard deviation	11.24 (0.77)	10.08 (0.54)	1.66 (0.06)	10.76 (0.96)
Minimum	197 (21)	166 (19)	29 (19)	306 (94)
Maximum	258 (24)	207 (22)	42 (20)	362 (99)
Mode/Median	217/228 (23/23)	172/181 (21/21)	32/32 (19/19)	336/330 (96/96)
Intervals > 250 ms	6	0	0	298
Audio Messages				
Frames Sent	17934	18014	18038	18020
Msgs(frames) Lost	4783 (4783)	4538 (4538)	0 (0)	2 (20)
Mean Frames Lost	15.94	15.08	0.00	0.07
Max Frames Lost	31	31	0	20
Video FPS				
Mean	13.26	23.35	29.96	13.78
Standard deviation	0.63	1.20	0.23	0.48
Minimum	10	11	28	12
Maximum	15	26	31	15
Mode/Median	13/13	23/23	30/30	14/14
Gaps	5022	1991	11	4882
Video Latency: Delivered				
Mean	261.37	220.65	72.34	295.14
Standard deviation	12.52	10.87	1.56	2.81
Minimum	231	204	70	290
Maximum	314	244	81	208
Mode/Median	271/262	208/220	71/72	294/295
Intervals > 250 ms	223	0	0	298
Video Messages				
Frames Sent	8966	9008	9019	9008
Frames Lost	4998	1982	0	4867
Mean Frames Lost	16.66	6.58	0.00	16.17
Max Frames Lost	20	9	0	17

Table 3-2: Summary of Capacity Constraint Demonstration #2

3.3.3. Access Constraints

3.3.3.1. Definitions of Access Constraints

A *structural access constraint* exists for $(m_S, b_S) \in OP_S$ if there is any hop k such that:

$$\left(1 \leq MA_k^{min}\, p_{s,k}\right) \vee \left(1 \leq PP_k\, p_{s,k}\right)$$

Structural access constraints exist because of limits in the number of packets that can be handled by some physical component of the network path. In particular, in the relation above, $p_{s,k}$ is the realized packet rate on hop k. A structural access constraint exists for $(m_S, b_S) \in OP_S$ if $p_{s,k}$ is too high given the value of MA_k^{min}, the minimum possible medium access time at the link server on hop k, or PP_k, the packet processing time at the node server for hop k. Physical network changes are required to relieve structural access constraints. Fortunately, structural access constraints are rare. With most current network technologies (*e.g.*, Ethernet, token rings, FDDI, T1 lines, *etc.*), MA_k^{min} is close to zero, so structural access constraints rarely occur as a result of minimum possible medium access. Furthermore, with current router technology and the packet rates generated by most video conferencing systems, structural access constraints rarely occur as a result of packet processing time.

A *congestion access constraint* exists for $(m_S, b_S) \in OP_S$ if the following holds for any hop k in the path.

$$\left(1 - U_{other,k}^{link} \leq MA_k\, p_{s,k}\right) \vee \left(1 - U_{other,k}^{node} \leq PP_k\, p_{s,k}\right) \qquad (3\text{-}5)$$

The expression $(1 - U_{other,k}^{link})$ describes the link server utilization available to service stream s. The expression $MA_k p_{s,k}$ describes the link server utilization required to acquire control of the medium for the offered packet rate. If this utilization exceeds the available utilization, the link server is saturated. In other words, it takes too long to acquire control of the medium for each packet given the incoming packet rate. The expression $(1 - U_{other,k}^{node})$ describes the node server utilization available to service stream s. The expression $PP_k p_{s,k}$ describes the node server utilization required to process packets at the offered packet rate. If this utilization exceeds the available node server utilization, the node server is saturated. In this case, it takes too long to process each packet given the incoming packet rate. Both of these situations result from the stream packet rate, regardless of the stream bit rate. Since the conference does not control

MA_k, $U^{link}_{other,k}$, or $U^{node}_{other,k}$, the only way for the conference to address the constraint is to select an operating point that generates a lower packet rate on hop *k*.

We say $(m_s, b_s) \in OP_s$ is *access constrained* if it has a structural or congestion access constraint. Addressing an access constraint requires lowering the number of packets transmitted, even if the resulting packets contain a larger number of bits per packet. Reductions to the stream bit rate, if not accompanied by reductions in the stream message rate, do not relieve access constraints. In addition, it may be possible to sustain a previously unsustainable bit rate by using a lower packet rate to address the access constraint. Figure 3-36 graphically illustrates a congestion access constraint. All operating points within the area labeled *Congestion Access Constraint* have congestion access constraints and cannot be successfully used by the conference.

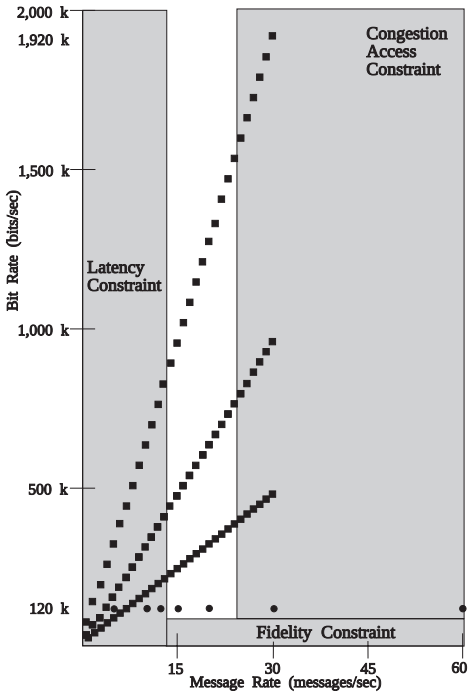


Figure 3-36: Congestion Access Constraint

The set of operating points that have congestion access constraints is a superset of the set of operating points with structural access constraints, so we say stream *s* is *fundamentally access constrained* if the following relation holds.

$$\forall (m_s, b_s) \in OP_s, \exists k \in \{1..h\}\ s.t.,$$
$$\left(1 - U^{link}_{other,k} \le MA_k\, p_{s,k}\right) \vee \left(1 - U^{node}_{other,k} \le PP_k\, p_{s,k}\right)$$

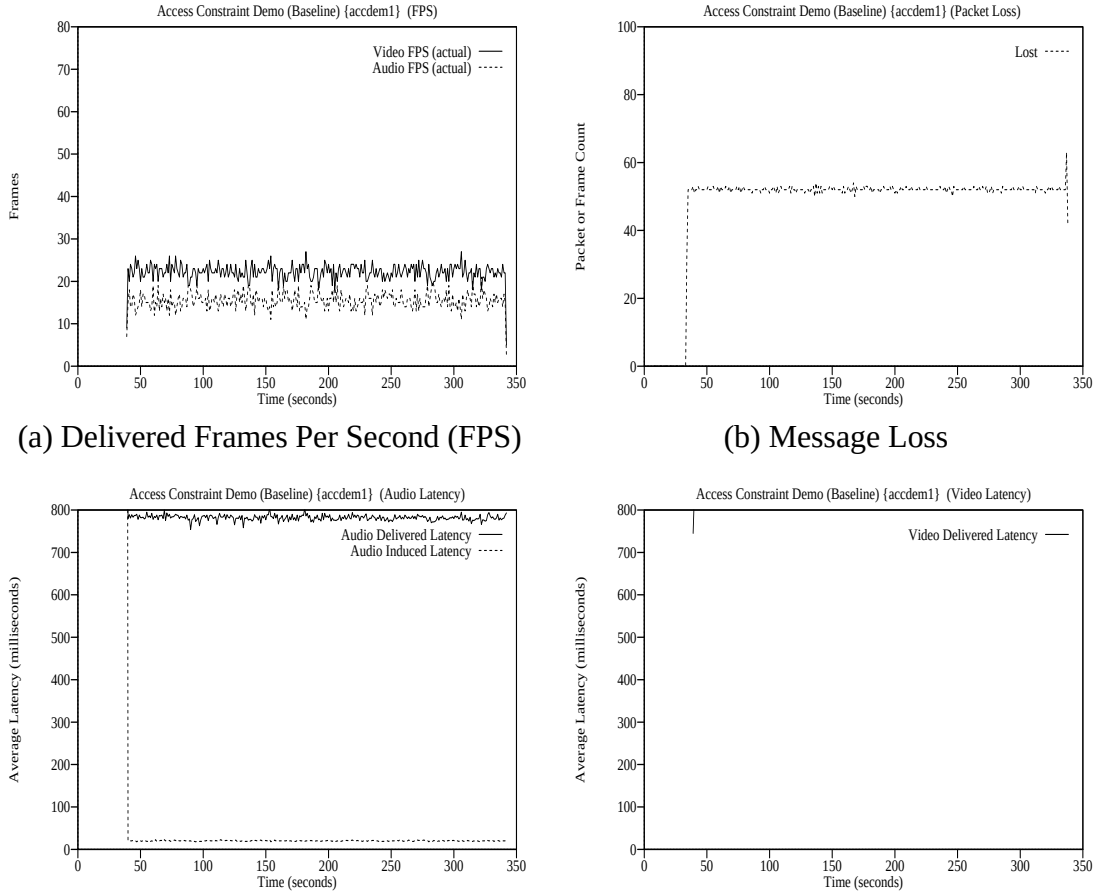
When a fundamental access constraint exists, the transmission strategy cannot address the constraint by manipulating a single stream *s* since no operating point can be successful. It is possible that the conference may still be able to relieve the constraint by manipulating multiple media streams in concert.

3.3.3.2. Demonstrations of Access Constraint Effects

It is easy to create an access constraint on shared medium networks. In the examples that follow, we again use the network shown in Figure 3-27. In these experiments, we removed the modified routing code that created capacity constraints in the previous set of demonstrations and router *R1* can process the full bit rate of the aggregate audio and video streams. Since the MTU for the token ring segments is large, the realization of the message rate on each hop gives $m_s = p_{s,k}$ for all hops for both streams. To generate the access constraint, we attached synthetic traffic generators to the middle token ring segment. These traffic generators generate a heavy traffic load on the middle token ring segment and result in competition for free tokens on the segment. Router *R1* must compete for tokens to forward the audio and video messages across the middle hop to router *R2*. The average delay required to acquire a free token is 24 *ms*, as measured by the IBM Trace and Performance token ring monitor program, so $MA_2 = 24$ *ms*.

In the first demonstration, audio and video use the operating points (60, 120k) and (30, 1920k), respectively (see Figure 3-4). Audio is congestion access constrained, since $MA_2 \times p_{a,2} = (0.024) \times 60 = 1.44 > 1$. The video stream is access constrained due to the presence of the audio stream. More specifically, $U_{other,k}^{link}$ for the video stream includes the utilization require to satisfy the audio stream. Since the utilization required to service the audio stream is greater than 1, there is no excess utilization for processing the video stream. The competition for buffers at *R1* between the audio and video streams also affects delivery of the video stream. Figure 3-37 shows the results for the first demonstration conference. Part (a) shows that the delivered frame rates for both audio and video are very poor. Ideally, the conference would deliver audio at 60 frames per second and video at 30, but neither stream reaches the desired values. Audio delivery is particularly poor. There are over 13,000 audio gaps (Table 3-3) and the resulting audio is unintelligible. Part (b) shows there is heavy message loss. As a result of the access constraint, the packets arriving at *R1* consume all the buffers and many packets are dropped at *R1*. The queue of waiting packets at *R1* also results in stream latencies far greater than our maximum latency guideline of 250 *ms*.

The average audio latency is 780 *ms* (Table 3-3 and part (c)) and the average video latency is 843 *ms* and off the top of the chart in part (d). Overall, the access constraint on the second hop leads to a conference with very poor quality.



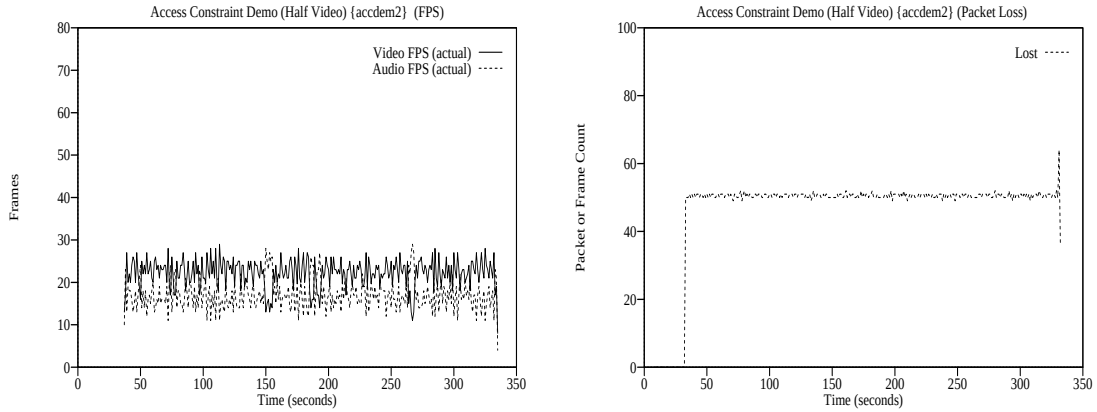
(a) Delivered Frames Per Second (FPS) (b) Message Loss

(c) Audio Latency (d) Video Latency

Figure 3-37: Access Constraint - MA_2 = 24 *ms* (Figure 3-27) - Baseline

Figure 3-38 shows the results of a conference using the audio and video operating points of (60, 120k) and (30, 960k), respectively. The conference scales back the video bit rate to try to address the network congestion. This conferences uses the same audio operating point as in Figure 3-37, but the video stream now uses the medium quality coding scheme. The medium quality coding reduces the video bit rate to half the high quality rate, but the video message rate stays the same with 1 frame per message. The change in video bit rate does not change the access constraint relations in any way. In particular, for audio $MA_2 \times p_{a,2} = (0.024) \times 60 = 1.44 > 1$, just as with the previous demonstration. The capacity constraint relations change, as discussed in the capacity constraint demonstrations, but since the conference is not

capacity constrained, the change is irrelevant. Figure 3-38 part (a) shows that there is essentially no change in the delivered frame rate for either streams. There are over 12,000 audio gaps (Table 3-3) and audio is still unintelligible. Loss is still extreme (part (b)) and the stream latencies (parts (c) and (d)) are still excessive (752 *ms* for audio and 805 *ms* for video; Table 3-3). Changing the video bit rate gave no relief to the access constraint, even though the aggregate bit rate of the conference was reduced by half. The implication is that a transmission scheme relying solely on video scaling is completely ineffective in this environment.



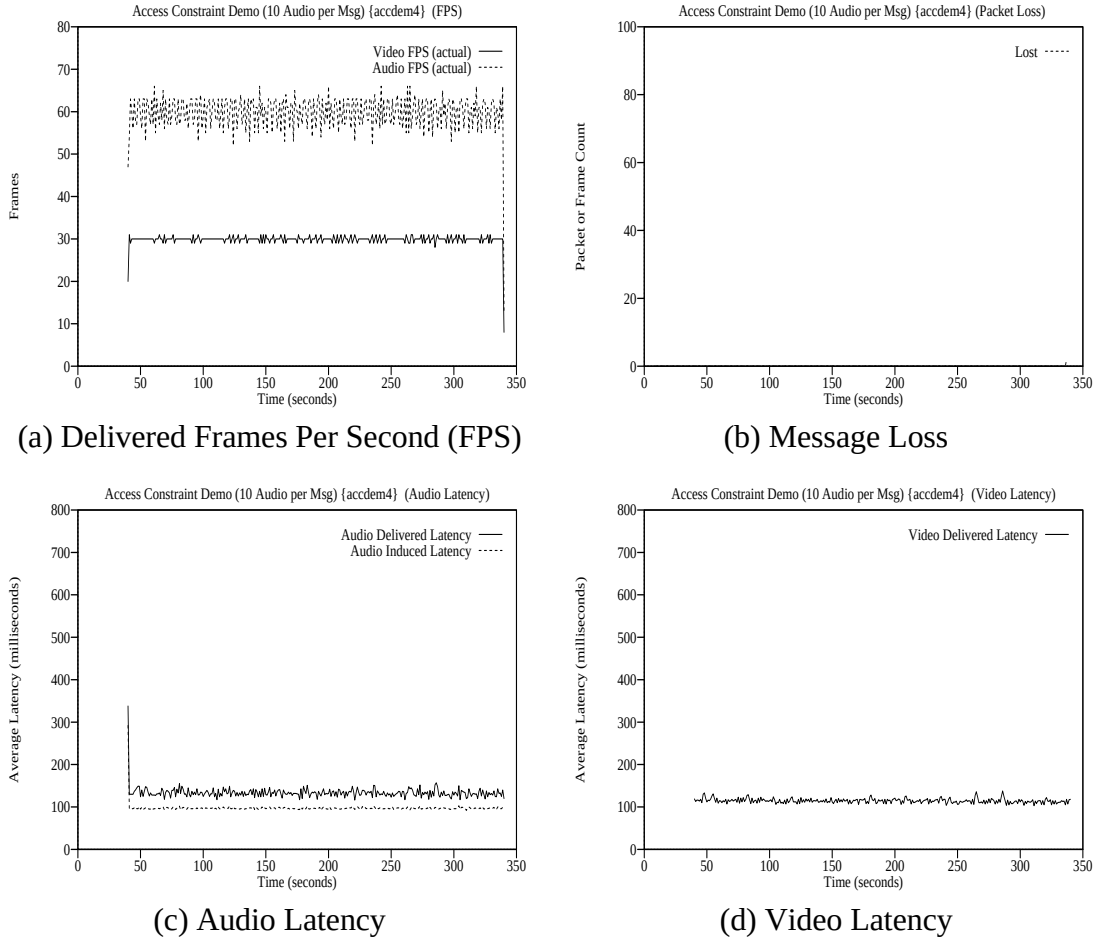
(a) Delivered Frames Per Second (FPS) (b) Message Loss

(c) Audio Latency (d) Video Latency

Figure 3-38: Access Constraint - MA_2 = 24 *ms* (Figure 3-27) - Medium Quality Video

Figure 3-39 shows the results of the conference when the audio and video operating points are (6, 120k) and (30, 1920k), respectively. This choice of operating points has exactly the same bit rate as the operating points (60, 120k) and (30, 1920k) used in the first access constraint demonstration (*i.e.*, Figure 3-37), but reduces the audio message rate to a tenth the message rate of the first demonstration by packaging 10 audio

frames in every message. This change in audio operating point changes the access constraint relation for audio, so now $MA_2 \times p_{a,2} = (0.024) \times 6 = 0.14 < 1$ and the audio stream is no longer access constrained.



(a) Delivered Frames Per Second (FPS)

(b) Message Loss

(c) Audio Latency

(d) Video Latency

Figure 3-39: Access Constraint - MA_2 = 24 *ms* (Figure 3-27) - Audio 10 Frames/Msg

Figure 3-39 part (a) shows that the change in operating points dramatically improves the delivered audio and video frame rates. The regular variation in the received audio frame rate is an artifact of summing audio frames over one second intervals. Since each message carries 10 audio frames, the graphed number varies depending on which interval is assigned a particular set of 10 audio frames. Audio playback is unaffected by this artifact. Audio fidelity is good. The average audio frame rate is 59.78 out of a possible 60 and there are only 17 audio gaps (Table 3-3). Video fidelity is also good. The conference uses the high quality video encoding and video is played at an average frame rate of 29.89. There is very little loss (only 2 video frames; Table 3-3), even though the same number of bits are transmitted in Figure 3-39 as in Figure 3-37. The

delivered audio stream latency is radically reduced, even though the induced audio latency increases to about 100 milliseconds as a result of the audio frame packaging (part (c)). Even with the induced latency, audio latency is only 133 *ms* (Table 3-3). Video latency is also reduced (part (d)). The lower latency in both streams is due to the elimination of the queueing delay in *R1*. *R1* is no longer constrained by the competition for tokens on the second hop. Packaging 10 audio frames per message leads to a conference that fully delivers the maximum bit rate available to the conference (Figure 3-4).

The implications of these demonstrations are significant. First, in its purest form, video scaling does nothing to address access constraints because the focus is on reducing the stream bit rate. We claim access constraints are common on existing LANs and video scaling is not well suited to address these constraints. Second, addressing access constraints via packaging significantly improves the conference quality. Conference fidelity and latency are improved and network loss is reduced. Finally, networks with large MTUs offer more flexibility in packaging due to the potentially large packet sizes and consequently offer more packaging options, which implies more flexibility in dealing with access constraints.

The existence of access constraints is well known [13, 65]. For example, router manufacturers routinely report bit rate throughput and packet rate throughput for routers. However, the recognition of the effect of access constraints on video conference quality and the dramatic potential improvements in quality possible through directly addressing access constraints is not generally acknowledged. Congestion within LANs is often considered solely a capacity issue, with competition for medium control viewed simply as a reduction in the potential capacity of the link. While there is value in this view, it hides the tremendous impact of message and packet rates. In addition, in the case of video scaling transmission schemes, viewing congestion as solely a capacity problem may lead to unnecessary reductions in conference fidelity (*e.g.*, by lowering the video bit rate) when packaging can resolve the congestion problem with no fidelity reductions (*e.g.*, consider Figure 3-39).

Access demo	Baseline	Medium Video	10 Audio
Audio FPS			
Mean	15.51	17.07	59.78
Standard deviation	2.01	3.55	4.27
Minimum	2	4	13
Maximum	21	29	66
Mode/Median	16/15	16/17	63/61
Gaps	13520	12845	17
Audio Latency: Delivered (Induced)			
Mean	781.83 (20.05)	752.25 (19.17)	133.01 (96.81)
Standard deviation	6.36 (0.93)	6.51 (0.64)	7.83 (1.95)
Minimum	754 (18)	724 (18)	116 (92)
Maximum	804 (23)	767 (21)	157 (103)
Mode/Median	781/782 (20/20)	751/753 (19/19)	131/133 (96/96)
Intervals > 250 ms	303	298	0
Audio Messages			
Frames Sent	18247	17944	18000
Msgs(frames) Lost	13505 (13505)	12831 (12831)	0 (0)
Mean Frames Lost	44.28	42.77	0.00
Max Frames Lost	49	51	0
Video FPS			
Mean	22.20	22.17	29.89
Standard deviation	1.91	3.32	1.40
Minimum	5	8	8
Maximum	27	29	31
Mode/Median	22/22	23/23	30/30
Gaps	2369	2345	11
Video Latency: Delivered			
Mean	843.54	805.04	114.55
Standard deviation	6.26	6.78	5.52
Minimum	816	780	103
Maximum	864	825	138
Mode/Median	844/844	805/805	115/115
Intervals > 250 ms	303	298	0
Video Messages			
Frames Sent	9123	8972	9001
Frames Lost	2360	2336	2
Mean Frames Lost	7.74	7.79	0.01
Max Frames Lost	16	18	2

Table 3-3: Access Constraint Demonstration Summary

Addressing access constraints via packaging is not a panacea. There are clearly access constraints that cannot be solved by any packaging scheme (*e.g.*, the worst case time to acquire control of a 16 megabit/second token ring is theoretically about 2.5 seconds [48]) and fidelity constraints limit the acceptable amount of induced latency. Nevertheless, we claim there are many congested network environments where packaging is a viable reaction to access constraints. Packaging and media scaling together can adapt to both capacity and access constraints. A transmission control scheme utilizing both techniques is likely to be more successful than one relying on only one.

3.3.3.3. Fragmentation and Access Constraints

In the previous demonstrations of access constraints on a token ring network, there is no message fragmentation since the MTU of each token ring hop is larger than any message produced by the conference. Fragmentation may exacerbate access constraints by producing a number of packets for every generated message. The generated packet rate on each hop is at least as high as the message rate and may be much higher (*i.e.*, $m_s \leq p_{s,k}$). Since hop k is access constrained if relation 3-5 holds, for given values of MA_k and PP_k, large values for $p_{s,k}$ are more likely to cause an access constraint than small values.

For example, suppose in Figure 3-27 that the effective MTU for the middle token ring segment is 1,500 bytes rather than 17,800. This situation sometimes occurs when default MTU values are used to configure routers (*i.e.*, some router software uses a default MTU of 1,500 bytes for both token rings and Ethernets). Let $MA_2 = 24$ *ms* as in the previous access constraint demonstrations. When the MTU is 17,800, the video stream is not access constrained for operating point (30, 1920k) since $MA_2 \times p_{v,2} = (0.024) \times 30 = 0.72 < 1$. However, when the effective MTU is 1,500, router *R1* must fragment each video message into $\left\lceil \frac{8{,}000}{1{,}500} \right\rceil = 6$ packets (see the realization in Figure 3-13). Now, $MA_2 \times p_{v,2} = (0.024) \times 180 = 4.32 > 1$ and the video stream is access constrained.

This phenomenon occurs not only because of medium access time, but also because of packet processing time, PP_k. Suppose there is no competition for control of the medium on hop 2, but that the packet processing time at *R2* is 10 *ms* (*i.e.*, $PP_2 = 10$ *ms*). With an MTU of 17,800, the packet rate on the second hop for the video

operating point (30, 1920k) is 30, so $PP_2 \times p_{v,2} = (0.010) \times 30 = 0.3 < 1$ and there is no access constraint at *R2*. When the MTU is 1,500, $PP_2 \times p_{v,2} = (0.010) \times 180 = 1.80 > 1$ and *R2* is access constrained.

When MTU sizes are small relative to media frame sizes, such as with the 8,000 byte video frames and 1,500 byte MTU in this example, packaging alone cannot address the access constraint. In this case, the only way to significantly lower the packet rate on the constrained hop is to lower the media stream bit rate to the point where media frames can be packed into a smaller number of packets. Since video frames are typically much larger than audio frames, video is more likely to experience access constraints resulting from fragmentation. Video bit rate scaling is thus sometimes effective against access constraints on networks with small MTUs, such as Ethernet. Reducing the bit rate of the video stream eventually has the secondary effect of reducing the generated packet rate, perhaps to the point where the access constraint is relieved. However, later chapters show that a transmission control strategy combining scaling and packaging can be even more effective addressing access constraints, even on networks where fragmentation occurs.

Fragmentation also has some effect on capacity constraints, but the effects are typically minimal. Fragmentation changes the visit counts at a hop by increasing the value of $p_{s,k}$, as follows:

$$V_{s,k} = \frac{p_{s,k}}{m_s}$$

However, since there is no significant increase in the stream bit rate as result of fragmentation (typically only the extra network protocol headers associated with each new packet resulting from fragmentation) and since the increase in visit count is offset by a proportional reduction in the transmission time of the packet (either internal data movement in the node or transmission time on the link), relations involving the overall stream bit rates do not change. Thus relation 3-4 still describes capacity constraints and we can evaluate the capacity constraint using only the aggregate stream bit rate b_s without regard to the generated packet rates.

Fragmentation often results in more buffers being consumed in routers than when no fragmentation occurs. For example, some routers use fixed size buffers. In these routers, the increase in packets caused by fragmentation directly increases the number

of buffers used in the router. Since more buffers are used in the router, the router is more likely to experience buffer shortages and drop packets. Fragmentation may also magnify the effect of packet loss on the conference fidelity. The network service cannot deliver a message to the conferencing application unless all fragments of the message arrive at the destination. If any fragments are lost, the network service cannot deliver the message and any frames carried in the message are lost. For example, suppose video frames are fragmented into 6 packets. If one packet in six is lost because of network congestion, it is possible no messages will be delivered to the application and no video played.

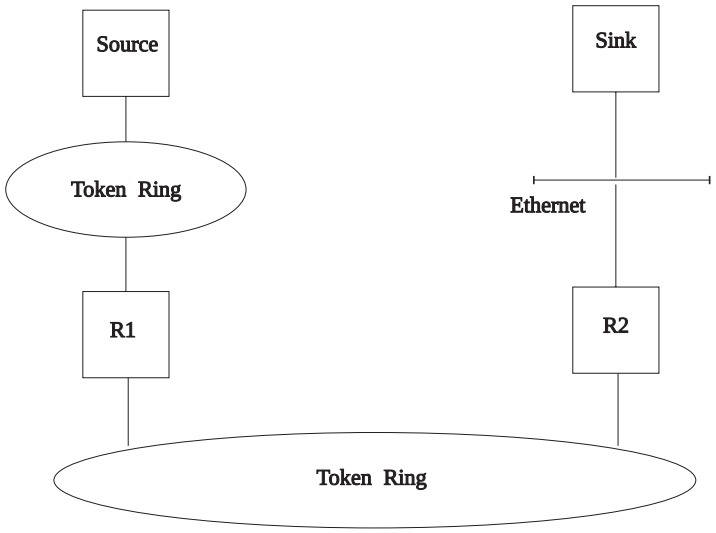


Figure 3-40: Token ring - Token Ring - Ethernet Network Path

The overhead and complexity of dealing with varying sized packets as a message crosses the network has led some to suggest fragmenting all messages at the source using the smallest MTU in the network path [65, 106]. This prevents intermediate nodes from having to deal with fragmentation and thus potentially improves their performance. Unfortunately, when the network segments can become congested independently, this strategy may unnecessarily expose the video conference to constraints that could otherwise be avoided. For example, consider the network in Figure 3-40. The source machine is connected to a token ring network with an MTU of 17,800. The middle token ring network also has an MTU of 17,800, while the Ethernet segment has an MTU of 1,500. Suppose, as with our earlier examples, that MA_2 = 24 *ms* and that the video operating point is (30, 1920k). If fragmentation is done as needed on a hop by hop basis, the MTU on the middle token ring network will

be 17,800 and the video stream will not be access constrained (since $MA_2 \times p_{v,2}$ = (0.024) × 30 = 0.72 < 1). However, if we use the strategy of fragmenting to the lowest MTU at the source, the effective MTU on the second hop is 1,500 and the video stream experiences an access constraint on the second hop (*i.e.*, $MA_2 \times p_{v,2}$ = (0.024) × 180 = 4.32 > 1). Thus from a video conference perspective, it is best to fragment as late as possible along the network path in an attempt to avoid potential access constraints. There is also an argument for reassembling messages at intermediate hops to avoid access constraints [65, 62], but this strategy requires changing existing routers and so is not considered here.

3.3.4. Combination Constraints

Of course, an operating point may be both capacity and access constrained, as shown by the area labeled *Access and Capacity Constraint* in Figure 3-41.

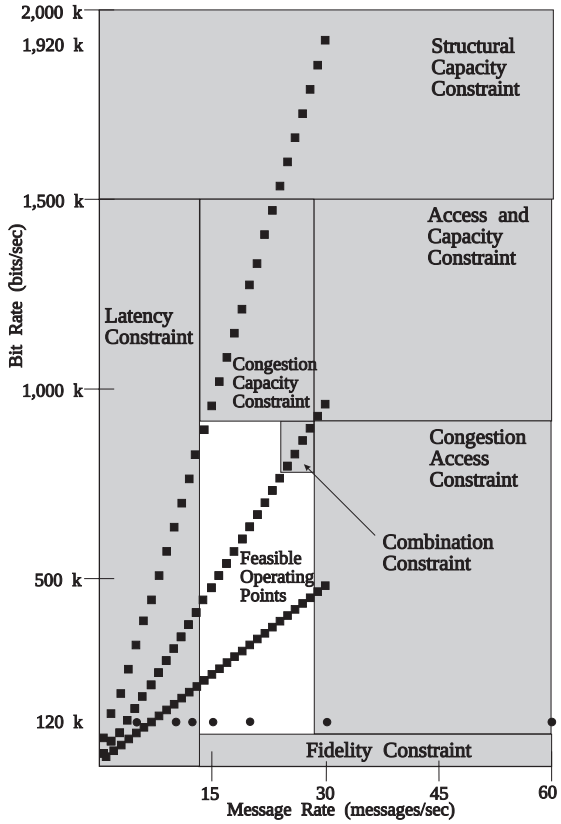


Figure 3-41: Exclusion of Operating Points due to Combination Constraints

Operating points may also be constrained through the combination of a particular bit and message rate. A *combination constraint* exists for $(m_S, b_S) \in OP_S$ if (m_S, b_S) is not access or capacity constrained (*i.e.*, relations 3-4 and 3-5 do not hold for (m_S, b_S)), but the following holds for some hop k in the path.

$$\left(1 - U_{other,k}^{link} \leq MA_k \, p_{s,k} + \frac{b_s}{r_k^{link}}\right) \vee \left(1 - U_{other,k}^{node} \leq PP_k \, p_{s,k} + \frac{b_s}{r_k^{node}}\right) \quad (3\text{-}6)$$

In this case, the combination of packet processing and bit processing time saturates some server in the path, while packet processing or bit processing alone does not. In this case, lowering either the bit or the message rate may sufficiently lower the service demand to relieve the constraint. In some cases, the conference must reduce both the bit and message rates to address the constraint. The operating points in the area labeled *Combination Constraint* in Figure 3-41 graphically represent those excluded from consideration because of combination constraints.

3.3.5. Summary of Constraints

A video conferencing application may be subject to a number of constraints when selecting operating points for the audio and video streams. Fidelity constraints and latency constraints are due to human perception. Capacity, access, and combination constraints are due to the physical constraints in the network and the amount of congestion present in the network. In the previous sections, we have described each of these constraints. Table 3-4 summarizes the symbols used in the constraint descriptions and Table 3-5 summarizes the relations describing the constraints.

Symbol	Meaning
f_s	The maximum frame rate associated with stream *s* (frames/second).
b_s	A bit rate associated with stream *s* (bits/second).
m_s	A message rate associated with stream *s* (messages/second).
(m_s, b_s)	An operating point for stream *s*.
OP_s	Set of all operating points for stream *s*.
F_s^{min}	Minimum acceptable bit rate for stream *s* (bits/second).
L_s^{max}	Maximum acceptable latency for stream *s* (seconds).
$L_s(t)$	Network latency for stream *s* at time *t* (seconds).
$Buf_s^{max}(t)$	Maximum buffering latency for stream *s* (*i.e.*, $\max(L_s^{max} - L_s(t), 0)$)
h	Number of hops in the path from conference source to sink.
$U_{other,k}^{link}$	Link server utilization resulting from all classes other than class *s*.
$U_{other,k}^{node}$	Node server utilization resulting from all classes other than class *s*.
r_k^{link}	Data transfer rate for the link server on hop *k* (bits/second).
r_k^{node}	Data transfer rate for the node server on hop *k* (bits/second).
$p_{s,k}$	Packet rate generated by class *s* on hop *k* (packets/second).
MA_k^{min}	Minimum possible time to acquire control of the medium associated with the link server on hop *k* (seconds).
MA_k	Average time to acquire control of the medium at the link server for hop *k* over some measurement interval (seconds).
PP_k	Average time to process one packet at the node server for hop *k* (seconds).

Table 3-4: Summary of Symbols

Constraint Type:	Constraint exists for $(m_s, b_s) \in OP_s$ if relation holds:
Fidelity	$b_s < F_s^{min}$
Latency	$m_s < \left\lceil \frac{f_s}{\lceil f_s Buf_s^{max}(t) \rceil} \right\rceil$ where $Buf_s^{max}(t) = \max(L_s^{\max} - L_s(t), 0)$
Structural Capacity	$\exists k \in \{1..h\},\ s.t.\ \left(1 \le \frac{b_s}{r_k^{link}}\right) \vee \left(1 \le \frac{b_s}{r_k^{node}}\right)$
Congestion Capacity	$\exists k \in \{1..h\},\ s.t.\ \left(1 - U_{other,k}^{link} \le \frac{b_s}{r_k^{link}}\right) \vee \left(1 - U_{other,k}^{node} \le \frac{b_s}{r_k^{node}}\right)$
Structural Access	$\exists k \in \{1..h\},\ s.t.\ \left(1 \le MA_k^{min} p_{s,k}\right) \vee \left(1 \le PP_k p_{s,k}\right)$
Congestion Access	$\exists k \in \{1..h\},\ s.t.\ \left(1 - U_{other,k}^{link} \le MA_k p_{s,k}\right) \vee \left(1 - U_{other,k}^{node} \le PP_k p_{s,k}\right)$
Combination	Not congestion access or congested capacity constrained, but $\exists k \in \{1..h\},\ s.t.\ \left(1 - U_{other,k}^{link} \le MA_k p_{s,k} + \frac{b_s}{r_k^{link}}\right) \vee \left(1 - U_{other,k}^{node} \le PP_k p_{s,k} + \frac{b_s}{r_k^{node}}\right)$

Table 3-5: Summary of Constraint Relations

3.3.6. Dynamic Nature of Congestion Constraints

Capacity, access, and combination constraints are dynamic. The constraints vary because of changes in other traffic loads at the servers (*i.e.*, the influence of $U_{other,k}^{link}$ and $U_{other,k}^{node}$) or in the average time required to gain control of network links (*i.e.*, the influence of changes in MA_k). During different intervals, different sets of operating points may be excluded from consideration due to congestion constraints. Graphically (see Figure 3-41), the set of points excluded by capacity congestion constraints expands and contracts vertically. As capacity congestion increases, the areas labeled *Congestion Capacity Constraint*, *Access and Capacity Constraint*, and *Combination Constraint* may expand by moving the bottom boundary lines toward the *x*-axis. As capacity congestion decreases, these areas shrink by moving the bottom boundary line away from the *x*-axis. Similarly, as access congestion increases, the areas labeled

Congestion Access Constraint, *Access and Capacity Constraint*, and *Combination Constraint* may expand by moving the leftmost boundary line of the areas toward the *y*-axis. As congestion due to access constraints decreases, the leftmost boundary lines move away from the *y*-axis. This implies the conferencing application must periodically evaluate the set of operating points to determine if particular operating points are constrained or unconstrained given the current network conditions.

Let $U_{other,k}^{link}(t)$ and $U_{other,k}^{node}(t)$ be the average utilization at the link and node servers, respectively, for all classes $c \in C$ and $c \neq s$ at hop k at time t. Let $MA_k(t)$ be the average medium access time at the link server on hop k at time t. The set of candidate operating points for stream s at time t, $COP_s(t)$, is the set of operating points that can be sustained given the levels of network congestion at time t.

$$COP_s(t) = \{(m_s, b_s) | (m_s, b_s) \in OP_s, \forall k \in \{1..h\},$$
$$\left(1 - U_{other,k}^{link}(t) > MA_k(t)\, p_{s,k} + \frac{b_s}{r_k^{link}}\right) \wedge \left(1 - U_{other,k}^{node}(t) > PP_k\, p_{s,k} + \frac{b_s}{r_k^{node}}\right)\Bigg\} \quad (3\text{-}7)$$

$COP_s(t)$ excludes all operating points that are capacity, access, or combination constrained at time t. If $COP_s(t)$ is empty, then no operating point for stream s can be successful in the current environment and stream s is *fundamentally constrained.*

Congestion affects not only $COP_s(t)$, but also $POP_s(t)$ (*i.e.*, the set of perceptual operating points; see equation 3-1). Congestion caused by either access or capacity constraints typically increases the network latency, which causes the rightmost boundary of the *Latency Constraint* area (Figure 3-41) to move farther from the *y*-axis, as described in the earlier discussion of perceptual constraints. Similarly, decreases in either type of congestion may cause the rightmost boundary of the *Latency Constraint* area to move toward the *y*-axis. Increasing congestion thus has the double effect of eliminating points from the top or right and simultaneously from the left.

The remainder of this chapter discusses how a video conferencing application can identify a set of operating points that are in both $COP_s(t)$ and $POP_s(t)$. The conference sender must be able to identify these operating points using only the definitions of the operating points and feedback from the conference receiver. The conferencing application must periodically determine this set of operating points over the life of the conference. Since congestion constraints change over time, the set of

operating points that are in both $COP_s(t)$ and $POP_s(t)$ also changes over time and the video conference must adapt to these changes in order to maintain a quality conference.

3.4. Feasible Operating Points

Recall that the key problem with transmission control is to choose a *feasible operating point*, which is a sustainable operating point with adequate fidelity and latency. For stream *s* at time *t*, the set of feasible operating points, $FOP_s(t)$, is what remains after we remove all points from the set OP_s excluded because of perceptual, capacity, access, and combination constraints. The remaining points are those that provide an adequate quality conference and can be successfully delivered in the current network environment. $FOP_s(t)$ is thus the intersection of the perceptual (3-1) and candidate operating points (3-7).

$$FOP_s(t) = POP_s(t) \cap COP_s(t) \tag{3-8}$$

Figure 3-41 graphically represents $FOP_s(t)$ as the operating points in the area labeled *Feasible Operating Points*. The video conference application could select any operating point in this set and deliver an acceptable quality conference (although some points in the set are clearly more desirable than others). $FOP_s(t)$ is a finite set of points and we fundamentally assume at least one point exists in $FOP_s(t)$ for all media streams (*i.e.*, the conference is not fundamentally constrained). If $FOP_s(t)$ is null, there are no sustainable operating points that provide adequate quality for media stream *s* and the transmission control algorithm cannot succeed; relief can only come from decreases in $U^{link}_{other,k}(t)$, $U^{node}_{other,k}(t)$, or $MA_k(t)$, which are beyond the control of the conference streams.[1]

3.4.1. A Superset of the Feasible Operating Points

[1]This is actually an over-simplication. Since we are considering streams independently, it may be that by scaling and packaging multiple conference streams together we may relieve the congestion. In later chapters, we demonstrate that controlling the audio and video streams independently, but adapting both at the same time can relieve constraints that cannot be relieved by adapting a single stream alone.

Ideally, we would directly compute $FOP_S(t)$ and select operating points based on this calculation. Unfortunately, to do this we would need to evaluate the constraint relations for each hop in the path. In current networks, this is not possible since there is too little information available at the conference endpoints to compute the results.

We can, however, identify useful subsets and supersets of $FOP_S(t)$. First, we consider a superset of $FOP_S(t)$. Points in this set may or may not be feasible. To identify this set, we assume that the link server on the first hop in the path is the bottleneck server. We do this because we can often directly measure parameters from the first hop (*i.e.*, at the conference sender) and compute constraint relations. We assume that the conference source node is capable of supporting any operating point available to the video conference, that the transmission rate of the first link is sufficient to carry the bit rate of any operating point, and that the node server on the first hop is capable of supporting any packet rate associated with the conference. These are all reasonable assumptions for well-built systems using existing LANs. With these assumptions, the first hop can only be a bottleneck if the medium access time is too long. In particular, the medium access time is too long for operating point (m_S, b_S) to be feasible if:

$$1 \le MA_1(t)\, p_{s,1} + \frac{b_s}{r_1^{link}}$$

$$1 - \frac{b_s}{r_1^{link}} \le MA_1(t)\, p_{s,1}$$

$$\frac{1}{p_{s,1}} - \frac{b_s}{p_{s,1} r_1^{link}} \le MA_1(t) \qquad \text{(3-11)}$$

We can restate equation (3-11) in a positive way by saying the operating point (m_S, b_S) is sustainable if the packet generation period (*i.e.*, $1 / p_{S,1}$) minus the transmission time of the packet (*i.e.*, $\frac{b_s}{p_{s,1} r_1^{link}}$) on the first network link is greater than the medium access time. That is,

$$\frac{1}{p_{s,1}} - \frac{b_s}{p_{s,1} r_1^{link}} > MA_1(t) \qquad \text{(3-12)}$$

We can describe a superset of $FOP_S(t)$, $SUP_S(t)$, by combining relation (3-12) with the earlier description of the perceptual operating points.

$$SUP_s(t) = POP_s(t) \cap \left\{ (m_s, b_s) \left| \frac{1}{p_{s,1}} - \frac{b_s}{p_{s,1} r_1^{link}} > MA_1(t) \right. \right\} \qquad (3\text{-}13)$$

We assume we know the MTU of the first network, so we can compute $p_{s,1}$ for every operating point in OP_S. We also assume we know the transmission rate of the first network, r_1^{link}, so we can solve equation 3-12 for $MA_1(t)$ *a priori* for every potential operating point. Recall that to determine $POP_S(t)$, we must know the stream latency (since we must know $Buf_s^{max}(t)$, which depends upon the latency $L_S(t)$; see equation 3-1). Since we can estimate the network latency, $L_S(t)$, using feedback from the conference destination and directly measure the medium access time, $MA_1(t)$, at the conference source, we can compute $SUP_S(t)$ at the conference source for some time t, using equation 3-13.

3.4.2. A Subset of the Feasible Operating Points

Identifying a subset of the feasible operating points is more difficult than identifying a superset. Let $m_s^{fb}(t)$ be the most recently delivered message rate for stream s at the conference destination and let $b_s^{fb}(t)$ be the most recently delivered bit rate at time t. The receiver computes both these values over the last feedback interval and returns the values to the sender via feedback. The subset of the feasible operating points consists of all operating points satisfying the perceptual constraints and with equivalent or lower message and bit rates than the delivered rates.

$$SUB_s(t) = POP_s(t) \cap \left\{ (m_s, b_s) \middle| m_s^{fb}(t) \geq m_s \wedge b_s^{fb}(t) \geq b_s \right\}$$

If all messages transmitted for stream s were delivered during the previous feedback interval, no server in the path is overloaded and all operating points with lower total service demand than the current operating point will work. It is possible other operating points are also feasible, but since we do not know the relative contribution of packet processing and data transfer time, we can only guarantee lower service demand by constraining both message and bit rates to below the delivered rates. $SUB_S(t)$ is thus a subset of $FOP_S(t)$ because there may be other combinations of bit and message rates with service demands lower than the delivered stream, but with either message or bit rates above the last successfully delivered rates. It is also possible that additional operating points with both message and bit rates exceeding those delivered in the previous period are also feasible. Since the delivered rates

cannot exceed the transmitted rates except over brief periods, using delivered rates to estimate feasible operating points is necessarily conservative. The network may be able to sustain operating points with higher service requirements, but the sender cannot determine which additional points, if any, are sustainable given only feedback from the current operating points.

$SUB_S(t)$ is the null set when some of the messages transmitted for stream *s* are not delivered to the conference destination. When the delivered message and bit rates are smaller than the transmitted message and bit rates, some packets were lost in the network or remain queued in the network, which implies some server in the path is overloaded. The actual packets dropped at an overloaded server is a function of the buffer management policies in the server, the workload levels, and the order packets arrive. Although we know the bottleneck server was able to process a message rate of m_s^{fb} and a bit rate of b_s^{fb} over the last feedback interval, there is no guarantee that the same rates will be delivered in the next interval, even if the workload of all other classes remains constant. When the overloaded server sheds excess workload, there is no guarantee that the discarded work will be apportioned fairly or consistently among the job classes at the server. Using end-to-end transmission control, we cannot know the degree to which a server is overloaded, so the only guaranteed subset of $FOP_S(t)$ when loss occurs is the empty set. Furthermore, the conference source cannot determine the type of congestion present since the symptoms of capacity or access constraints are the same: high delivery latency and packet loss. We know of no discriminator to distinguish between capacity and access constraints solely from information available at the conference endpoints. Finally, when packet loss occurs, the messages that are successfully delivered to the conference destination usually have high delivery latency, $L_S(t)$, often resulting in a small or null set $POP_S(t)$ (see equation 3-1). Thus, when the message and bit rates fed back from the destination match the sending rates, we use $SUB_S(t)$ as the subset of $FOP_S(t)$. When the fed back message or bit rates are lower than the transmitted rates, we use the empty set as the subset of $FOP_S(t)$. Unfortunately, both of these subsets are often excessively conservative.

3.4.3. An Estimate of the Feasible Operating Points

Using the superset and subset of $FOP_S(t)$, the conference application can limit the points that are considered when picking an operating point. We can eliminate points that are definitely not in $FOP_S(t)$ (*i.e.*, those not in $SUP_S(t)$). We can also identify

points definitely within $FOP_S(t)$ (*i.e.*, those points in $SUB_S(t)$). The difference between $SUB_S(t)$ and $SUP_S(t)$ contains the operating points that we are not sure about. A conservative strategy is to simply pick operating points from $SUB_S(t)$, provided the set is not empty. These points are guaranteed to provide a sustainable conference under the current network conditions and deliver adequate conference quality; however, choosing points exclusively from $SUB_S(t)$ may lead to unnecessarily conservative operating points. Such a strategy often does not operate at the highest possible quality given the network conditions because there may be sustainable operating points outside $SUB_S(t)$ that have higher fidelity (*e.g.*, higher bit rates or higher frame rates) and lower latency (*e.g.*, due to lower induced latencies). A more aggressive strategy is for the conference to search the points $SUP_S(t) - SUB_S(t)$ to better match the true $FOP_S(t)$.

Unfortunately, $SUP_S(t) - SUB_S(t)$ may contain many operating points. We would like to estimate a good place to begin the search so that the first guess is close to the upper boundary of $FOP_S(t)$ (*i.e.*, the boundary furthest from the origin). The basic heuristic of the transmission control algorithm is to try to match the message period (*i.e.*, the inverse of the message rate) to the residence time at the bottleneck server. In other words, we try to match the rate at which the conference introduces work into the network to the rate work is being processed at the bottleneck server. When the conference has a message period shorter than the residence time at the bottleneck server, the bottleneck will pace the stream and the interarrival times between messages at the destination are likely to match the residence time at the bottleneck. When the conference message period is longer than the residence time at the server, the interarrival times of the messages at the destination will likely match the original message period since the bottleneck is able to process work faster than work is introduced. We can guess if the current message rate is acceptable by observing the actual interarrival time and comparing the interarrival time to the message period. If the message period is less than the interarrival time, there is probably a bottleneck server in the path. If the message period is about the same as the interarrival time, messages are being delivered roughly at the transmitted rate and there probably is not a bottleneck.

The receiver measures the message interarrival time by calculating the average time between delivery of messages to the separation stage (see Figure 3-1). The average

message interarrival time for stream *s* at time *t* is $IT_S(t)$. We estimate $FOP_S(t)$ with the following:

$$EST_s(t) = SUP_s(t) \cap \left\{ (m_s, b_s) \;\middle|\; \frac{1}{m_s} \geq IT_s(t) \right\} \qquad (3\text{-}15)$$

$EST_S(t)$ takes a subset of the superset $SUP_S(t)$ as a better estimate of the actual set $FOP_S(t)$. $EST_S(t)$ considers only those operating points in $SUP_S(t)$ with message periods greater than the measured average interarrival times over the last feedback period. It is indeterminate whether $EST_S(t)$ is a superset or subset of $FOP_S(t)$, but empirical measurements show it is usually larger than $SUB_S(t)$, but smaller than $SUP_S(t)$. A heuristic transmission control algorithm can use $EST_S(t)$ as the basis for a search across the operating points. The next chapter describes such an algorithm.

3.4.4. Summary of Operating Points Sets

We have now described seven sets of operating points for a given stream *s*. Table 3-6 lists these sets. OP_S is the set of operating points for stream *s* that the conferencing application can produce. Operating points are described as combinations of message rates and bit rates. The set of perceptual operating points, $POP_S(t)$, is a subset of OP_S that provides acceptable perceptual quality at a time *t* during the conference. In particular, $POP_S(t)$ defines the operating points with acceptable latency and fidelity (bit rates) at time *t*. $COP_S(t)$ is a subset of OP_S that includes the operating points that are sustainable under the network conditions at time *t*. The set of feasible operating points, $FOP_S(t)$, is the intersection of $POP_S(t)$ and $COP_S(t)$. These operating points provide acceptable perceptual quality and are sustainable. Ideally, we would select operating points directly from $FOP_S(t)$. Unfortunately, we cannot directly compute $FOP_S(t)$ because an end-to-end transmission control scheme cannot determine enough information about the characteristics and load of the network path. Since we cannot compute $FOP_S(t)$, we estimate it instead.

$SUP_S(t)$ is a superset of $FOP_S(t)$ created using information acquired by the conference sender from the first hop in the path. $SUB_S(t)$ is a subset of $FOP_S(t)$ derived from information collected by the conference receiver and returned to the conference sender in feedback messages. The receiver calculates the received message rate for stream *s* over the last feedback interval, $m_s^{fb}(t)$, and the received bit rate for stream *s* over the last feedback interval, $b_s^{fb}(t)$. The receiver also calculates the average message

interarrival time for stream *s* over the last feedback interval, $IT_s(t)$, and also transmits this value to the conference sender in the feedback messages. Since there may be many operating points in the set $SUP_s(t) - SUB_s(t)$, we use $IT_s(t)$ to heuristically determine an estimated set of feasible operating points, $EST_s(t)$. $EST_s(t)$ is the basis for our transmission control algorithm described in the next chapter.

Set	Definition
OP_s	$OP_s = \{(m_s, b_s)\|$ the video conferencing application can produce a bit rate of b_s bits/second for stream *s* and transmit this bit rate with a message rate of m_s messages/second.$\}$
$POP_s(t)$	$POP_s(t) = \left\{(m_s, b_s) \middle\| (m_s, b_s) \in OP_s, 1/m_s \le Buf_s^{max}(t) \wedge b_s \ge F_s^{min}\right\}$
$COP_s(t)$	$COP_s(t) = \{(m_s, b_s) \middle\| (m_s, b_s) \in OP_s, \forall k \in \{1..h\}, \left(1 - U_{other,k}^{link}(t) > MA_k(t)\, p_{s,k} + \frac{b_s}{r_k^{link}}\right) \wedge \left(1 - U_{other,k}^{node}(t) > PP_k\, p_{s,k} + \frac{b_s}{r_k^{node}}\right)\}$
$FOP_s(t)$	$FOP_s(t) = POP_s(t) \cap COP_s(t)$
$SUB_s(t)$	$SUB_s(t) = POP_s(t) \cap \left\{(m_s, b_s) \middle\| m_s^{fb}(t) \ge m_s \wedge b_s^{fb}(t) \ge b_s\right\}$
$SUP_s(t)$	$SUP_s(t) = POP_s(t) \cap \left\{(m_s, b_s) \middle\| \frac{1}{p_{s,1}} - \frac{b_s}{p_{s,1} r_1^{link}} > MA_1(t)\right\}$
$EST_s(t)$	$EST_s(t) = SUP_s(t) \cap \left\{(m_s, b_s) \middle\| \frac{1}{m_s} \ge IT_s(t)\right\}$

Table 3-6: Summary of Operating Point Sets

3.5. Summary of the Transmission Control Framework

This chapter describes a framework for transmission control of audio and video streams. The capabilities of a video conferencing system are characterized by a set of operating points for each media stream. The operating points describe the potential combinations of message and bit rates available for that stream. At any point, the conferencing application uses one operating point from this set to generate and transmit the media stream. The set of operating points provides a concise abstraction of the conferencing systems capabilities from the perspective of transmission control. The task of the transmission control policy is to periodically select operating points for each media stream so that the conference quality is preserved under the current network conditions.

The task of selecting an operating point is complicated by the fact that the desirability of an operating point changes over time. At any point in the conference, a particular operating point may or may not lead to delivery of a high fidelity, low latency media stream. We have described how human perception may constrain the choice of operating points used for a conference due to latency or fidelity concerns. The physical network or network congestion may also eliminate operating points from consideration by making operating points unsustainable. These perceptual and transmission constraints may be either static or dynamic.

Network congestion often causes the most severe constraints on operating points. The effects of network congestion are explained using a simple queueing model. We have defined and demonstrated capacity and access constraints and shown the relationship of bit and message rates to these constraints. Capacity constraints are caused by either (1) limited network bandwidth on the transmission link, or (2) internal data movement time at forwarding nodes. Capacity constraints depend only on the stream bit rate and are not affected by the message rate or resulting packet rate. Capacity constraints must be addressed by reductions in the stream bit rate. Access constraints are caused by (1) medium access times when transmitting across shared-medium networks, or (2) packet processing time at a node server. Access constraints must be addressed by message rate reductions. Fragmentation may exacerbate access constraints.

Both capacity and access constraints are dynamic and may occur separately or in combination. We claim access constraints are more common than capacity constraints

on current local area networks, but most video conferencing transmission control algorithms either do nothing to address congestion or rely solely on bit rate reductions. To be successful, a transmission control must address both types of congestion and adapt to changing network conditions. End-to-end transmission control is not a panacea and network conditions exist where no end-to-end control algorithm can succeed, but we claim that adaptive end-to-end transmission control can deliver a quality conference under many adverse network conditions. We claim a transmission control scheme can do this by identifying a set of feasible operating points that provide adequate conference quality and are sustainable under the current network conditions. Identifying the set of feasible operating points is essentially a search problem over a subset of the entire set of operating points. This chapter discusses some heuristics for identifying a good candidate subset of operating points to initiate the search and the next chapter describes an algorithm that uses these heuristics to implement a transmission control algorithm.

Chapter IV
Recent Success Algorithm

The transmission control framework presented in Chapter 3 characterizes the set of operating points that are both sustainable and that will lead to an acceptable quality conference. Although the set of feasible operating points, $FOP_S(t)$, cannot be exactly computed at run-time, it can be estimated using $SUP_S(t)$, $SUB_S(t)$, and $EST_S(t)$. This chapter describes an algorithm that uses estimates of $FOP_S(t)$ to adaptively find feasible operating points when congestion makes the current operating point infeasible. The chapter also describes one possible implementation of the algorithm in C.

4.1. Conceptual Introduction to the *Recent Success* Algorithm

A transmission control algorithm manages the generation and transmission of the media streams in a video conference by selecting the operating point used for each stream. Ideally, the control algorithm would periodically compute the set of feasible operating points $FOP_S(t)$ for each media stream and select operating points directly from this set. Unfortunately, as we described in Chapter 3, an end-to-end transmission control scheme cannot directly compute $FOP_S(t)$ because the conference endpoints do not have enough information about the state of the network. We can estimate $FOP_S(t)$, but there may be many operating points that we are not sure about; they may or may not be in the feasible set. Furthermore, the conference endpoints cannot directly determine if the network is capacity or access constrained (*i.e.*, the externally observable symptoms are the same for both types of constraints). As a result, if the current operating point is infeasible we cannot definitively know whether to lower the message rate, the bit rate, or both. Since we cannot be sure which operating points are feasible, we must guess. We have created an algorithm called *Recent Success* that implements a set of strategies for making good guesses about how to adapt the media streams.

The *Recent Success* (RS) algorithm essentially implements a heuristic search of the operating points associated with a stream. The purpose of this search is to find an

operating point that provides acceptable quality and can be sustained under the current network conditions. When the conference streams are controlled independently, the algorithm is executed independently for each stream and maintains state about each stream. Part of the RS algorithm is implemented in the video conferencing application at the media source and part is implemented in the conferencing application on the receiving node. The part of the RS algorithm implemented at the source node selects the operating point for the stream. The portion of RS implemented at the receiving node simply collects information about the received message rates, message latency, and message interarrival times for the stream and periodically sends this information to the source node in *feedback* messages.

When using the *Recent Success* algorithm, the sending node takes the following steps. First, the sender periodically receives feedback from the receiver about a particular stream. Although each feedback message may carry feedback data about multiple media streams, we logically treat each stream separately. Second, the sender uses the feedback data for a particular media stream *s* to evaluate whether or not the current operating point is a member of $EST_s(t)$. Next, the sender executes transitions in a state machine to select a new operating point. The transitions taken may differ depending on whether or not the current operating point is in $EST_s(t)$. Finally, the sender sets the new stream operating point to the operating point resulting from the state transition. We describe these steps in more detail later in this chapter.

The basic idea behind RS is to heuristically characterize the network as either primarily capacity or primarily access constrained and select operating points on the basis of this characterization. Over time, RS may change the characterization based on how well particular operating points deliver the stream to the receiver.

In this section, we introduce the *Recent Success* algorithm by giving an example using the operating points for a particular system (see Figure 4-1). The intent of the example is to give the reader some intuition about how RS operates. We discuss the details of the algorithm later in this chapter.

4.1.1. A Simple Example

Suppose the current video operating point for a conference is (30, 1920k) (see Figure 4-1) and that RS is currently characterizing the network as primarily capacity constrained. Suppose that when the next feedback message arrives from the

conference receiver, we determine that (30, 1920k) is not in $EST_V(t)$. In other words, (30, 1920k) no longer appears feasible. Since RS believes the network is capacity constrained, RS reduces the bit rate for the video stream by changing the operating point to (30, 960k). This changes the video encoding scheme from high to medium quality. We say that the stream is *retreating* or in a *retreat state* when the current operating point is infeasible and the transmission control algorithm must reduce either the message or bit rate of the stream. After the video stream retreats to (30, 960k), RS changes the classification of the network from capacity to access constrained. If on the next feedback, (30, 960k) is also found to be infeasible, RS now reduces the video message rate by selecting a lower frame rate for the medium quality encoding and sets the characterization of the network back to capacity constrained. Thus, when a stream is retreating, RS follows a "stairstep" course, alternating between lowering the stream bit rate (when the network is believed to be capacity constrained) and lowering the stream message rate (when the network is believed to be access constrained). RS continues this stairstep course towards the origin over several feedback intervals until it either finds a successful operating point or there are no more operating points left to try.

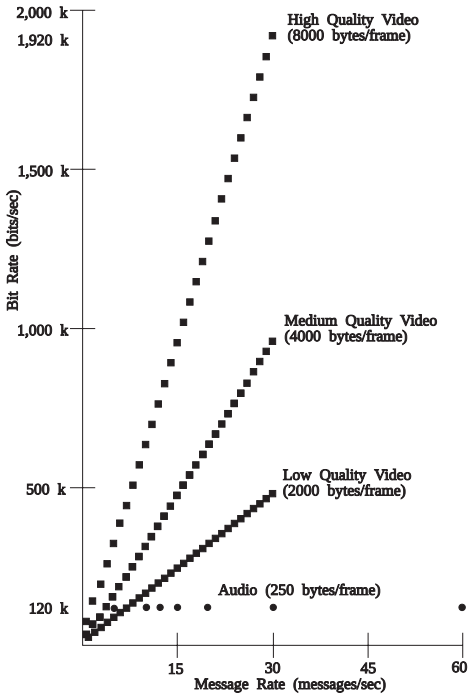


Figure 4-1: Sample Operating Point Description

Suppose that RS finally finds a sustainable operating point at (15, 240k). In the system described in Figure 4-1, this operating point corresponds to generating 15 video frames per second using the low quality video encoding and sending each frame in a separate message. Over the next few feedback intervals, the operating point (15, 240k) is consistently found sustainable (*i.e.*, it is consistently a member of $EST_V(t)$). Since this operating point does not offer the best possible video quality and since the network has sustained the operating point over several intervals, RS attempts to improve the quality of conference by selecting a better video operating point. Suppose that RS is currently characterizing the network as capacity constrained. The current operating point has been stable for several feedback intervals, so RS makes the tentative hypothesis that the capacity constraint in the network has been relieved. To test this hypothesis, RS selects a new operating point (15, 480k) that increases the video bit rate (in this case by changing to medium quality video rather than low quality). If after another stable period, (15, 480k) is also found to be sustainable, RS continues to increase the bit rate of the stream and selects (15, 960k) as the operating point. We say the stream is *probing* or is in a *probe state* when the current operating point remains feasible for a period of time and the transmission control algorithm attempts to improve the quality of the stream by increasing the stream message or bit rate. Probing continues along a single dimension (*i.e.*, increasing either the message or bit rate) until there are no more operating points along that dimension or the probe fails.

If a probe fails (*i.e.*, the newly selected operating point proves infeasible), RS returns to the previously sustainable operating point and changes the classification of the network. Networks previously considered access constrained are now considered capacity constrained and vice versa. If the previous operating point again proves sustainable, the next probe will be along a different dimension in Figure 4-1. For example, if the probe fails when increasing the bit rate, the next probe attempts to increase the message rate. Thus, as network conditions improve, RS gradually migrates over time from operating points close to the origin to operating points with higher message and bit rates.

In the following sections, we describe the *Recent Success* algorithm in more detail. We start by describing the feedback data collected by the conference receiver. The receiver periodically returns this data to the sender in feedback messages. The feedback data is the basis for decisions about when and which operating point to

select. We then describe how RS determines whether or not the current operating point is feasible (*i.e.*, if (m_S, b_S) is in $EST_S(t)$). Next, we describe the states, transitions, events, and actions in the RS state machine. We initially describe a simplified version of the RS state machine. After describing the basic state machine, we discuss some additional heuristics and optimizations to the state machine. Finally, we describe an implementation of *Recent Success* in C.

4.1.2. Measuring Feedback Data

Collecting the data for feedback is straight-forward. For each media stream, the receiver counts the number of messages received during a feedback interval, the total latency for all messages received during the feedback interval (*i.e.*, the sum of the latency for all messages arriving during the interval), and the total interarrival time for all messages received during the interval (*i.e.*, the sum of the interarrival time measured for each arriving message). The receiver also keeps the minimum latency for any received message (even those messages delivered outside the current feedback interval). When the feedback interval ends, the receiver sends a feedback message to the sender with the average message latency (total latency divided by the number of messages received during the interval), the average message interarrival time (total interarrival time divided by the number of messages received during the interval), and the minimum latency of any message received so far.[1] Since most conferences have media streams in each direction, feedback messages are usually "piggy-backed" on one of the conference media streams.

The length of the feedback interval is configurable. In the experiments in this dissertation, we have chosen a feedback interval of 200 milliseconds for both the audio and video streams[2]. We chose this interval as a compromise between transmission

---

[1]Counting incoming messages and computing the message interarrival times is straight-foward since they can be determined solely from information available at the receiving node. Computing the message latencies is more complicated because the the sender and the receiver do not have shared clocks. We defer our discussion of computing the latencies until later in the chapter.

[2]Each media stream could in principle have a different feedback interval, but we have chosen to use the same interval length for both audio and video.

control overhead and responsiveness. The transmission control algorithm is executed every time the sender receives a feedback message. Shorter feedback intervals cause an increase in the overhead associated with transmission control. The transmission control algorithms used in this dissertation (*e.g., Recent Success*) do not consume many resources (the routines have low memory requirements and execute less than 100 lines of C code), but we wish to avoid unnecessary overhead. On the other hand, feedback must be frequent enough to react to fast building congestion. In particular, in this dissertation we have chosen a maximum acceptable latency of 250 milliseconds. We have chosen a feedback interval less than this maximum value to give the transmission control algorithm an opportunity to address congestion before the congestion causes excessive latencies.

4.1.3. Determining if the Current Operating Point is in $EST_S(t)$

Each time the conference sender receives feedback from the conference receiver, the sender executes the RS algorithm. The first step in this algorithm is to determine if the current operating point is a member of $EST_S(t)$ based on the average message interarrival time at the conference destination (*i.e.*, $IT_S(t)$) and the network latency of the stream messages (*i.e.*, $L_S(t)$). These values are supplied directly from the feedback data.

The current operating point (m_S, b_S) is in $EST_S(t)$ only if it satisfies several conditions. Recall the definition of $EST_S(t)$ from Chapter 3.

$$EST_s(t) = POP_s(t) \cap \left\{ (m_s, b_s) \left| \frac{1}{p_{s,1}} - \frac{b_s}{p_{s,1} r_1^{link}} > MA_1(t) \right. \right\} \cap \left\{ (m_s, b_s) \left| \frac{1}{m_s} \geq IT_s(t) \right. \right\}$$

where:

$$POP_s(t) = \left\{ (m_s, b_s) | (m_s, b_s) \in OP_s, 1/m_s \leq Buf_s^{max}(t) \wedge b_s \geq F_s^{min} \right\}$$

To be in $EST_S(t)$, (m_S, b_S) must first satisfy the minimum acceptable bit rate for the stream. In other words, the condition $b_S \geq F_s^{min}$ must hold. Second, the induced latency must be acceptable given some predetermined, stream-specific maximum latency. In particular, the stream message period must be less than the maximum acceptable induced latency given the current network latency. This requirement is expressed as $1/m_S \leq Buf_s^{max}(t)$, where $Buf_s^{max}(t)$ is the maximum acceptable induced

latency and is equal to $L_s^{max} - L_S(t)$. If (m_S, b_S) satisfies these first two conditions, the operating point is a member of $POP_S(t)$[3].

Third, the packet period on the first hop ($1/p_{S,1}$) minus the transmission time on the first hop ($1/p_{s,1}r_1^{link}$) must be greater than the medium access time on the first hop ($MA_1(t)$). This condition is the second clause in the definition of $EST_S(t)$ above and determines if the offered message rate is sustainable on the first hop in the path. Finally, the current message period must be greater than or equal to the message interarrival time measured at the destination (*i.e.*, $1/m_S \geq IT_S(t)$)[4].

If the current operating point satisfies all the conditions above, it is in $EST_S(t)$ and we say the operating point is a *success*. If the operating point is not in $EST_S(t)$, the operating point is a *failure*. If RS determines an operating point is a success, we say a *success event* has occurred. If RS determines an operating point is a failure, we say a *failure event* has occurred. Later in this chapter, we describe how the RS state machines makes transitions on success and failure events.

4.1.4. *Recent Success* States

The *Recent Success* algorithm uses a state machine to keep track of the state of the media stream and the current classification of the network (*i.e.*, either capacity or access constrained). Transitions in the state machine select the next operating point for the stream. Figure 4-2 gives a simplified description of the *Recent Success* state machine.

---

[3]The evaluation of the minimum bit rate condition can usually be done at system definition and need not be evaluated on each feedback message. Operating points that do not satisfy the minimum bit rates for acceptable fidelity are simply omitted from the definition of the operating points.

[4]In practice, we allow some tolerance in the comparison of the message period to the interarrival time and to the induced latency. We discuss this tolerance in more detail later in the chapter.

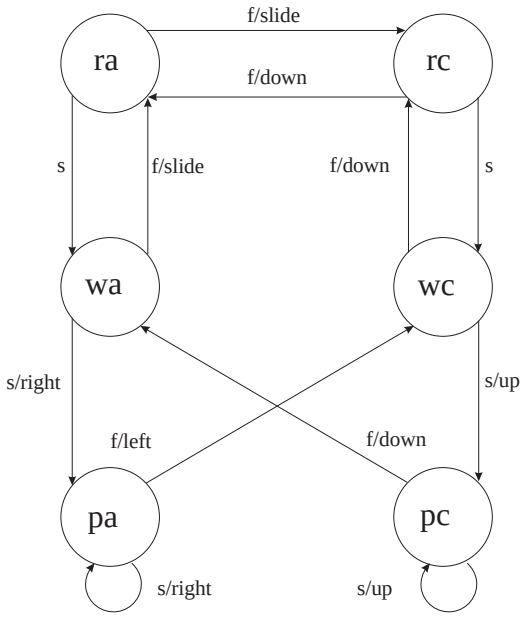


Figure 4-2: Recent Success State Diagram

There are six states in the state machine: **ra**, **wa**, **pa,** **rc**, **wc**, and **pc**. The state machine is in exactly one state at any point in the conference and we refer to this state as the *current state*. RS controls each media stream (*e.g.*, audio and video) independently, so each stream has its own current state. We say that stream *s* is *in state x* if the state machine associated with stream *s* has current state $x \in$ {**ra**, **wa**, **pa,** **rc**, **wc**, **pc}.**

There are three classes of states in the state machine. The *retreat* states (**ra** and **rc**) are states where the current operating point is infeasible. If stream *s* is in a retreat state, the RS algorithm attempts to lower the message or bit rate associated with *s* by selecting a new operating point for *s*. When stream *s* is in a retreat state, RS "retreats" from operating points with higher message or bit rates to those with lower rates. The *wait* states (**wa** and **wc**) are states where the current operating point is feasible and the network may be at equilibrium. If stream *s* is in a wait state, the RS algorithm makes no changes to the current operating point for *s*. When stream *s* is in a wait state, RS "waits" for a period of relatively stable performance before attempting to improve the quality of the stream. The *probe* states (**pa** and **pc**) are states where the network may be able to support operating points with higher message or bit rates than the current operating point. When stream *s* is in a probe state, RS "probes" the network by tentatively trying new operating points with higher message or bit rates than the current operating point to determine if the network can support the higher rates.

The stream state also captures the current characterization of the primary network constraint. RS believes the network is capacity constrained whenever the stream is in **rc**, **wc**, or **pc**. RS believes the network is access constrained when the stream is in **ra**, **wa**, or **pa**.

Each stream has a natural state. The stream enters its natural state when the algorithm starts or when the current network conditions are supporting the stream's maximum message and bit rates. When congestion occurs after a long period of stability, the natural state provides RS a hint on the first direction to look when searching for a new operating point.

The natural state for audio is **wa** (**wa** characterizes the network as access constrained). Audio streams usually have operating points with relatively low bit rates. Since capacity constraints are more likely to affect streams with high bit rates, audio is relatively insensitive to capacity constraints. On the other hand, audio streams often support high message rates to reduce the induced latency for the audio stream. The high message rates may make audio sensitive to access constraints. Making the natural state of the audio stream **wa** gives RS a hint on how to react if congestion occurs when audio is operating at its maximum operating point. The **wa** state characterizes the network as access constrained, so RS reduces the message rate if the current operating point is infeasible.

The natural state for video is **wc**. Video may have very large bit rates and thus may be very sensitive to capacity constraints (**wc** characterizes the network as capacity constrained). The natural reaction to network congestion when video is operating at its maximum message and bit rates is to reduce the video bit rate (*e.g.*, by changing the coding scheme).

4.1.5. State Transitions

State transitions are made based on whether or not the current operating point is in $EST_s(t)$. Each transition in Figure 4-2 is annotated with the event triggering the transition and the action associated with the event. Success events are labeled with ***s*** and failure events are labeled with ***f***. Actions are labeled as ***up***, ***down***, ***left***, ***right***, or ***slide*** and correspond to moving the operating point in the indicated direction on the transmission framework diagram (see Figure 4-1). The figure uses Mealy machine notation and has the event name, a slash, and the action associated with the event

(*e.g.*, ***s/right*** means move ***right*** when an ***s*** event occurs). If a transition has no associated action, only the event is shown. We discuss the specific meanings of the actions in more detail below.

4.1.5.1. Success Transitions

The action associated with a success event differs based on the state. If the stream is currently in a retreat state (**ra** and **rc**), the success event indicates that the stream may have adapted to current network conditions. The stream state switches to the wait state for the current network classification (*i.e.*, **ra** goes to **wa** and **rc** goes to **wc**), but the stream operating point is not changed.

If the stream is currently in a wait state (**wa** and **wc**) and feedback indicates the operating point is a success, the *Recent Success* algorithm attempts to improve the quality of the stream. It does this by increasing the message or bit rate of the stream and moving to one of the probe states. The idea here is that the stream has been successfully operating at the current operating point for some time and it is time to probe to determine if the network can now support (due perhaps to decreased congestion in the network) some higher quality operating point.

Note that $EST_S(t)$ rarely includes operating points with higher message rates than the current operating point. This is because the interarrival times of the messages at the destination cannot be less than the message period for any substantial length of time. The message period will eventually pace the interarrival time if there is no network congestion. This means that RS must actually attempt to transmit at a higher message rate to conclusively determine whether the higher message rate is feasible.

In state **wa**, improving the quality of the stream means increasing the message rate to the next available operating point along the message rate axis. In this case, RS believes the network is access constrained, but the network has been stable for a period of time. We tentatively hypothesize that the access constraint has subsided and test this hypothesis by using a higher message rate. This corresponds to moving ***right*** along the message rate axis on the transmission framework diagram. For example, suppose the audio stream is currently operating at (15, 120k) (see Figure 4-1). Moving right corresponds to moving the operating point to (20, 120k). In the conferencing system described in Figure 4-1, this change translates to packaging 3

audio frames per message instead of 4. It increases the audio message rate and decreases the induced latency in the audio stream.

For video, in the system described in Figure 4-1 moving ***right*** implies increasing the message (and frame) rate of video along the current coding scheme line. Geometrically, this implies both "right" and "up" on the diagram, but we choose to describe this as moving ***right*** since there is no change in coding scheme.

In state **wc**, improving the quality of the stream means increasing the bit rate of the stream. The rationale is similar to that with **wa**. RS has characterized the network as capacity constrained, but the network has been stable for some time. We tentatively hypothesize that the capacity constraint has subsided or is less severe. To test this hypothesis, we increase the bit rate of the stream. This corresponds to moving ***up*** along the bit rate axis on the transmission framework diagram. For example, suppose the video stream is currently using the medium quality encoding scheme and transmitting 15 messages per second (*i.e.*, operating point (15, 480k)). Moving up changes the video operating point to (15, 960k), which implies changing to the high quality encoding scheme.

Success in one of the probe states leads to additional increases in conference quality. These increases in conference quality mean moving ***right*** in **pa** to lower the induced latency by raising message rates or moving ***up*** in **pc** to give higher fidelity by increasing bit rates. These increases continue until either we reach the maximum operating point along a dimension or feedback shows a failure.

4.1.5.2. Failure Transitions

Failure events occur when feedback indicates the current operating point is not in $EST_s(t)$. As with success events, the actions taken on failure events are different depending upon the current state. Failures in the wait or retreat states cause the stream to begin or continue retreating along the message or bit rate axis. Moving ***left*** in the transmission framework diagram means lowering the message rate along the message rate axis (*e.g.*, moving the audio operating point from (20, 120k) to (15, 120k)). Moving ***left*** for video (at least in the system described in Figure 4-1), corresponds to moving towards the origin such that the message (and frame) rate

decreases, but the coding scheme stays the same (*e.g.*, moving from (15, 480k) to (14, 476k))[5].

A ***slide*** means moving left across perhaps multiple operating points to the first operating point with a message period greater than or equal to the interarrival times measured at the destination and returned by feedback. For example, suppose the returned interarrival time for the audio stream messages is 90 *ms* and the current operating point is (60, 120k). To get to the new operating point, the algorithm "slides" across several operating points to reach (10, 120k) that has a message period (100 *ms*) greater than the interarrival time. The effect is to move quickly from the current infeasible operating point to an operating point in $EST_s(t)$.

Moving ***down*** means lowering the bit rate (*e.g.*, changing the coding scheme to produce fewer bits per frame). In the system in Figure 4-1, moving video ***down*** means changing from one video encoding scheme to a scheme producing fewer bits per frame (*e.g.*, moving from (15, 960k) to (15, 480k)). In this particular system, changing the coding scheme is a fairly dramatic bit range change, but other systems may support more subtle changes (*e.g.*, using techniques such as quad-tree encoding [18]).

Failure events that occur during one of the probe states probably mean the stream has just crossed over the boundary of $FOP_s(t)$. The stream would not have reached the probe stage without success at a nearby operating point. If a failure occurs, the response then is to "undo" the most recent change in operating point by either moving the operating point ***left*** (when in **pa**) or ***down*** (when in **pc**). The classification of the network changes when a failure occurs in the probe states (see Figure 4-2). Networks previously considered access constrained are now considered capacity constrained and

---

[5]Another interpretation of moving ***left*** is to move towards the y-axis by reducing the video frame (and message) rate, but keeping approximately the same bit rate. This implements a "pure" ***left*** move in that the message rate is decreased, but the bit rate is essentially constant. For example, if the current operating point is (15, 480k) and uses the medium quality encoding, the operating point selected by a ***left*** move would be (7, 448k) and uses the high quality encoding. In the system described here, the video message rate is directly related to the video frame rate. We have chosen for perceptual reasons to stay within a given coding scheme for ***left*** and ***right*** moves to avoid the relatively large changes in frame rate associated with "pure" ***left*** and ***right*** moves.

vice versa. The rationale is that the failure of the most recent change in the probe state showed the stream is operating at an edge of $FOP_S(t)$ for either message or bit rates and further quality increases must come along the other dimension.

4.2. Enhancements and Extensions to the Basic State Machine

The previous section describes the basic *Recent Success* algorithm, but omits some details for clarity. This section describes these details.

4.2.1. Estimating Network Latencies without Synchronized Clocks

The *Recent Success* algorithm uses the message network latencies to evaluate $EST_S(t)$; however, measuring network latency is complicated because the conference sender and receiver do not have synchronized clocks. The sender sends a timestamp with each message that is the time that the message is passed to the network service. This time is based on the sender's clock. If the sender and receiver had synchronized clocks, the receiver could easily determine the network latency by subtracting the message timestamp from the current time when the message is received. However, when the clocks are unsynchronized, we do not know how to compare the time from the sender's clock to the time on the receiver's clock. Techniques exist for establishing synchronized clocks across a network, but these techniques tend to be complicated and expensive. Other end-to-end control schemes have described ways to eliminate the effects of clock skew using round-trip messages without the need for synchronized clocks [43].

RS takes an even simpler approach. Each message sent from the sender is tagged with a timestamp from the sender's clock. Without synchronizing with the sender, the receiver node subtracts this timestamp from the time of receipt based on the receiver's clock (since the clocks are not synchronized, the result could be negative). The receiver keeps track of the smallest difference ever measured between the transmit timestamp and the receipt timestamp. This minimum difference is subtracted from the newly calculated difference to give an estimate of the network latency.

The assumption behind this latency calculation heuristic is that since the conference is relatively long lived, it is likely that the network path is not congested throughout the life of the conference. Occasionally, at least one message will arrive with near the minimum possible transmission time across the path. This minimum becomes the basis

for measuring the latency due to congestion. Since latencies due to congestion are typically much larger than the uncongested transmission time, the estimated network latency approximately matches the actual latency.

For example, consider the audio stream described in Figure 4-1 and the network in Figure 4-3. If the network is unloaded, the minimum time to transmit the largest message available in the audio stream is in the 3-5 *ms* range. This assumes no delay in the routers and a 120,000 bit message. In this case, the transmission time on the token rings is approximately (120,000 / 16,000,000) × 3 = 2.25 *ms*. When the network is congested, on the other hand, latencies are measured in hundreds of milliseconds (see the capacity and access constraint examples in sections 3.3.2 through 3.3.4 of Chapter 3). Longer network paths (*i.e.*, more hops) may give higher minimum latencies, but the inaccuracies are blurred by the relative size of the congested latencies. This latency calculation heuristic gives a very cheap and reasonably accurate estimate of the actual network latency, particularly during congested periods when the estimate is most critical to the transmission control algorithm. If a more accurate estimate is required, techniques such as that described by Haas may be used [43], but the simple heuristic algorithm has proven adequate on several networks, as demonstrated in the next two chapters.

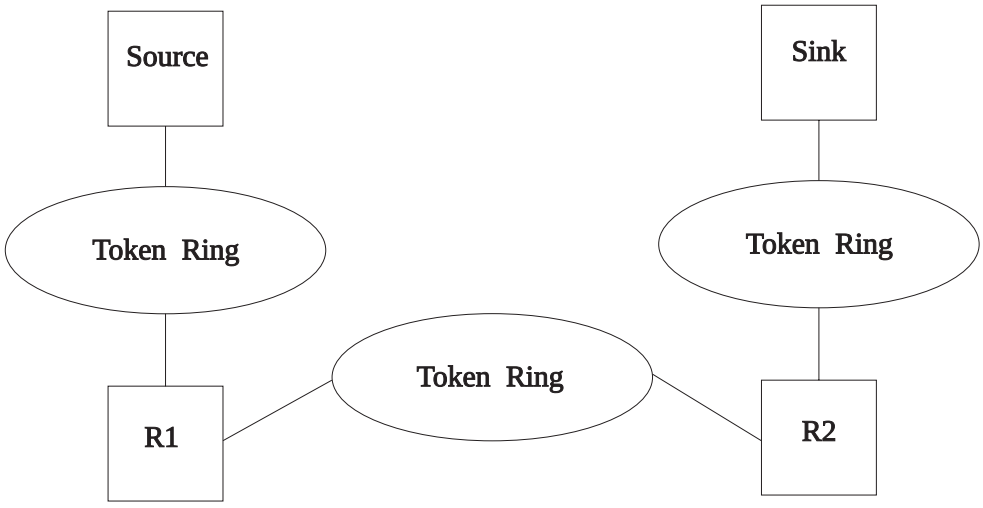


Figure 4-3: Sample Network Path

4.2.2. Determining EST_s without Medium Access Times

Chapter 3 described $EST_s(t)$ using the set $SUP_s(t)$. Calculating $SUP_s(t)$ requires that the transmission control algorithm know the medium access time on the first hop. In some systems (such as the ones used in this work), this value is available to the application, but in many systems, particularly when the transport layer software is separated from the application, this value is not directly available. Similarly, evaluating $SUP_s(t)$ also assumes the transmission speed of the first link is known. This value is sometimes available and sometimes not. We devise a new estimation of the feasible operating points, $\overline{EST_s}(t)$, for systems where MA_1 and r_1^{link} are not known.

$$\overline{EST_s}(t) = POP_s(t) \cap \left\{ (m_s, b_s) \;\middle|\; \frac{1}{m_s} \geq IT_s(t) \right\}$$

This calculation of $\overline{EST_s}(t)$ uses only the set of perceptual operating points, $POP_s(t)$, and the message interarrival time, $IT_s(t)$. We can calculate $POP_s(t)$ using only the average message latency and the average message interarrival time. Both of these values are returned via feedback, so no information must be retrieved from the local transport software. This modification may slightly enlarge the candidate operating point search space, but is applicable to a wider range of video conferencing systems. The algorithm presented later in this chapter uses this revised calculation of $\overline{EST_s}(t)$, as do all the experiments described in later chapters. For convenience, in the remainder of this paper, we refer to $EST_s(t)$ and $\overline{EST_s}(t)$ interchangeably.

4.2.3. State Thresholds and Ramps

It is undesirable for the adaptation algorithm to adapt too quickly to feedback. It is reasonable to require a certain period of stability at a given operating point before attempting to move to operating points with higher network requirements. For this reason, the states in the *Recent Success* algorithm have transition threshold values. The effect of these threshold values is to hold the stream in a particular state for a specified number of feedback intervals before enabling certain transitions.

There are two types of thresholds in *Recent Success*: *success thresholds* and *failure thresholds*. Success thresholds require that a specified number of consecutive feedback messages indicate the current operating point is successful before the success event can cause a transition out of the state. For example, we may choose to require 3

consecutive "successful" feedback messages before we allow a stream to move from **wa** to **pa.** Failure thresholds are analogous to success thresholds, but govern failure event transitions. The current implementation of RS has static thresholds, but adaptive threshold strategies are possible.

The magnitude of the success and failure thresholds govern how quickly RS adapts to changes in the network. Since thresholds are measured in terms of feedback intervals, the thresholds clearly must be set with a particular feedback interval in mind. Longer feedback intervals generally imply smaller thresholds and vice versa.

In practice, failure thresholds are often zero (*i.e.*, transitions are immediately enabled) to give fast response to building network congestion. In particular, we have chosen to use failure thresholds of 0 for the wait and retreat states for both the audio and video streams in the experiments in this dissertation. We chose these values so that the algorithm would react quickly to increases in congestion by lowering the conference message or bit rates. In our experiments, we have chosen to use a failure threshold of 1 for video and 2 for audio when in the probe states. We have adopted a general strategy of letting video adapt slightly faster than audio and these thresholds are consistent with that strategy. The motivation for this strategy is that we are willing to trade some volatility in the video stream for fast response to increasing or decreasing congestion, but we want more stable performance in the audio stream.

Success thresholds for retreat states and probe states are generally low (*e.g.*, 0-2 for feedback periods of 200 *ms*). For the experiments in this dissertation, we use success thresholds of 1 for video and 2 for audio when in the retreat or probe states. We chose the values for the same reason described above for the failure thresholds in the probe states.

Success thresholds for the wait states are generally larger than other thresholds. Low success thresholds in the wait states allow the stream to quickly probe for higher quality conferences while high wait state success thresholds lead to a slower changes in operating points. Low thresholds are useful with networks that are infrequently congested, while high thresholds are useful for networks with relatively frequent periods of congestion. In this dissertation, we use wait thresholds of 5 for both the audio and video streams. We chose these threshold values in a fairly arbitrary manner (5 feedback periods with 200 *ms* feedback intervals implies the operating point must be stable for about 1 second before considering the network in equilibrium). Although

we have had good success with these thresholds, we have not investigated the sensitivity of these parameters or made any attempt to find optimal values.

As demonstrated by our threshold settings, different streams may use different success and failure thresholds. This allows streams to react at different rates to changes in the network. Setting the thresholds differently imposes more stability on the slow moving streams by allowing other streams to adapt more quickly to changes in the network. For example, we may want the video stream quality to rapidly deteriorate in the face of congestion in order to preserve a desired level of audio quality. Of course, different thresholds allow different reactions only if there are multiple streams and the streams are not controlled in concert as a combined stream.

4.2.4. Allowing for Uncertainty and Small Inaccuracies

Feedback measurements are aggregates (*e.g.*, sums and averages) and are subject to some measurement noise. To allow room for error, *Recent Success* uses a few blurring thresholds. For example, there is a blurring threshold (typically 1-2 *ms*) that allows for some measurement error in the interarrival time measured at the destination when compared to the sender's message period. For example, the threshold may cause the algorithm to consider a measured interarrival time of 18 *ms* "close enough" to the sender's message period of 16.75 *ms*. The current implementation of *Recent Success* has a blurring threshold for the interarrival time mentioned above and for comparisons with measured network latency.

For the experiments in this dissertation, we have chosen to use a 25 *ms* blurring threshold for comparisons involving audio latency. We use a 50 *ms* blurring threshold when comparing video latency. We use a blurring threshold of 2 *ms* for interarrival time comparisons. We chose these values empirically and have not attempted to determine optimal values for these thresholds.

4.2.5. Boundary Conditions Transitions

The state transition diagram in Figure 4-2 does not show some of the boundary case transitions that are actually present in the current implementation of *Recent Success*.

In the retreat states (**ra** and **rc**), there is a minimum rate check that prevents the system from moving from **ra** to **rc** if the stream is already operating at the minimum

bit rate coding scheme. Similarly, if the stream is already operating at the lowest supported message rate, the transition from **rc** to **ra** will not occur.

If the stream enters the wait or probe states (for either access or capacity constrained networks) and the stream is at the maximum of the associated operating point axis (message rate for access constrained or bit rate for capacity constrained), there are transitions not shown on Figure 4-2 that go to the corresponding state under the other network classification. For example, if the video stream is in **wa,** but is operating at its maximum message rate, the stream state becomes **wc**.

If a stream reaches a wait or probe state and the stream is operating at its maximum message and bit rates (*i.e.*, its maximum operating point), the state of the stream is set to the natural state for that stream. This guarantees that after long periods of stability using its maximum operating point the stream responds in its natural direction at the first return of congestion.

4.2.6. Latency Heuristic

We may augment the "success test" for feedback messages (determining if the current operating point is in $EST_S(t)$) with a check on the latency trend of the stream. With this adjustment, the stream considers an operating point successful only if it is in $EST_S(t)$ and the network latency of the stream is constant (within a blurring threshold) or decreasing. The rationale for this heuristic is that the stream should not consider the network at equilibrium or probe when the network latency is increasing. Increasing latency likely means that congestion is rising in the network. Even if the current operating point is within $EST_S(t)$ (*i.e.*, the period and induced latency tests are passed), the increasing end-to-end latency implies that $EST_S(t)$ is shrinking and it is inappropriate to increase the message or bit rate of the stream.

4.2.7. Poor Video Heuristic

The video conferencing systems used in the experiments in this dissertation control the audio and video streams independently. We have discussed that it is possible to control the two streams in concert by combining the operating points of the two streams to form a single logical stream; however, we do not use this technique in this work. Nevertheless, it is sometimes desirable to favor one stream over another even when the streams are transmitted and controlled separately. All experiments using RS

in this dissertation have only one case where one stream is explicitly penalized in an attempt to improve the quality of the other. Specifically, if the delivered video frame rate drops below a specified threshold (*i.e.,* 20 frames per second in the experiments described here), the transmission control scheme sets the audio stream operating point to the minimum message rate in (t). The audio bit rate is not changed, since we assume the audio bit rate is inconsequential when compared to that of the video stream. We call this explicit favoring of video over audio the "poor video" heuristic. Our threshold choice of 20 frames per second is motivated by our desire to keep the video frame rate above 15 frames per second if possible, even if we must induce latency into the audio stream to accomplish this.

4.2.8. Implementing Pure Scaling and Pure Packaging with *Recent Success*

A transmission control scheme that uses pure scaling attempts to control transmission of a media stream solely by manipulating the stream's bit rate. Similarly, a control scheme using pure packaging controls transmission solely by manipulating the number of frames packaged per message. Pure scaling and pure packaging are extreme ends of the spectrum of control possible with *Recent Success*. Pure scaling limits RS to changes along the bit rate axis and pure packaging limits RS to changes along the message rate axis. If all operating points in OP_S have the same message rate, RS becomes a pure scaling algorithm. Similarly, if all operating points in OP_S have the same bit rate, RS becomes a pure packaging algorithm. If the operating points are such that an increase in message rate implies an increase in bit rate and vice versa, RS becomes in essence a temporal scaling algorithm. Thus, depending upon the operating points defined for the stream, RS can act as a spatial scaling algorithm, a temporal scaling algorithm, a packaging algorithm, or a two-dimensional algorithm (*i.e.,* based on the complete transmission control framework). We will exploit this flexibility in the following chapter to compare the performance of different control algorithms. This flexibility also hints that RS will outperform all single-dimension algorithms when evaluated under a diverse set of network environments.

4.3. Required Controls within Conferencing System

For any adaptive transmission control algorithm to work, the conferencing system must provide a set of "knobs" that control the media streams' message and bit rates. Typically, this implies control of the media frame generation rate, the media coding

scheme and/or sampling rate (which controls the bits per frame), and the number of frames packaged into a message. Frame rate control may be a feature of the video codec or may be simulated within the video conference application. Application control of the frame rate usually just implies ignoring selected frames produced by the codec. The codec must provide coding scheme and sampling rate controls. Here, we assume there is no overhead for dynamically changing coding schemes. If the overhead is significant the transmission control algorithm must be biased to consider these system-specific overheads when selecting operating points. The transport layer may provide packaging control, but more likely the application will control packaging by not passing a message to the transport interface until the desired number of frames are ready for transmission. We assume a message interface. Stream interfaces, such as TCP, isolate the application from packaging decisions, thus removing the packaging control from the application.

The actual mechanisms for setting the conference operating points are system-specific. In the implementation discussed in the next section, we have hidden the details of these mechanisms in the system-specific routine **`set_stream_OP`**. This routine takes a stream name and an operating point as parameters. The routine uses the system-specific mechanisms to configure the conferencing system to generate and transmit the stream according to the specified operating point.

4.4. An Implementation of *Recent Success*

This section describes an implementation of *Recent Success* in C for a video conferencing system built at the University of North Carolina at Chapel Hill. Listing 4-1 shows the data structures used by RS. Each media stream is described with a **`stream_defn`** structure. This structure contains the description of the stream operating points, the state of the RS algorithm for the stream, the index of the current stream packaging scheme, the index of the current coding scheme, and a set of threshold counters. Each operating point is described with an **`op_entry`** structure that describes the operating point message and bit rate. For convenience, the structure also contains the message period, calculated from the message rate, and the maximum network latency at which this operating point is feasible. We calculate the maximum feasible network latency at system startup by subtracting the induced latency associated with the operating point from the maximum acceptable stream latency, L_s^{max} (250 *ms* for this dissertation).

```
#define AUDIOFPS 60 /* max audio FPS */
#define VIDEOFPS 30 /* max video FPS */

typedef struct {
    unsigned int video_interarrival;
    unsigned int video_latency;
    unsigned int min_video_latency;
    unsigned int audio_interarrival;
    unsigned int audio_latency;
    unsigned int min_audio_latency;
} feedback_header;

typedef struct {
    int bitrate;    /* bits/sec     */
    int msgrate;    /* msg/sec      */
    int msgperiod;  /* millisec/msg */
    int maxlatency; /* max feasible */
                    /* net latency  */
} op_entry;

/***********************************
* States:
* ra - retreat, access
* wa - wait, access
* pa - probe, access
* rc - retreat, capacity
* wc - wait, capacity
* pa - probe, capacity
***********************************/
/* algorithm state */
typedef enum {ra, wa, pa, rc, wc, pc}
   alg_state;
```

```
/* operating points */
typedef struct {
   op_entry op[MAXP][MAXC];

   int pind;        /* packaging index  */
   int maxpkgind;   /* max pkg index    */
   int cind;        /* coding index     */
   int maxcodeind;  /* max coding index */

   boolean inEST;   /* OP in EST?   */
   int prevlatency; /* last latency */

   alg_state state; /* algorithm state */
   alg_state natural;/* "natural" state*/

   /* "retreat" state info */
   int rramp;       /* ramp counter    */
   int rthreshold;  /* state threshold */
   /* "wait" state info */
   int wramp;       /* ramp counter    */
   int wthreshold;  /* state threshold */
   /* "probe" state info */
   int pramp;       /* ramp counter    */
   int pthreshold;  /* state threshold */

   /* tolerances */
   int ITtolerance   /* interarrival    */
   int LATtolerance  /* latency         */
} stream_defn;

stream_defn audio_op;
stream_defn video_op;
```

Listing 4-1: *Recent Success* Data Structures

This implementation of the RS exploits the fact that the conferencing systems used in the remaining chapters of this dissertation support each potential coding scheme for each potential message rate. In other words, if the conferencing system supports a particular message rate, it can use any of the available coding schemes with that message rate. Thus, we can describe the set of operating points with a two-dimensional array. One dimension of the array corresponds to the supported message rates and is indexed by the variable **`pind`**. We use the C programming language numbering convention, so an index of 0 gives the lowest message rate supported and **`maxpkgind`** the highest. The second array dimension identifies the available coding schemes and is indexed by the variable **`cind`**. When **`cind`** has the value 0, the coding scheme generating the smallest number of bits per frame is in use; when **`cind`** has the

value **maxcodeind** the coding scheme generating the most bits per frame is in use. Thus, the operating point in use is identified by the packaging index, **pind**, and the coding index, **cind**. Figure 4-4 shows the conceptual view of the operating points array for the video stream from the system in Figure 4-1. In Figure 4-4, the contents of the operating point array are shown as particular (m_S, b_S) values. In actual practice, the array is more likely to contain system-specific information about how to set the generation, compression, and packaging parameters in the actual system to achieve the desired operating point.

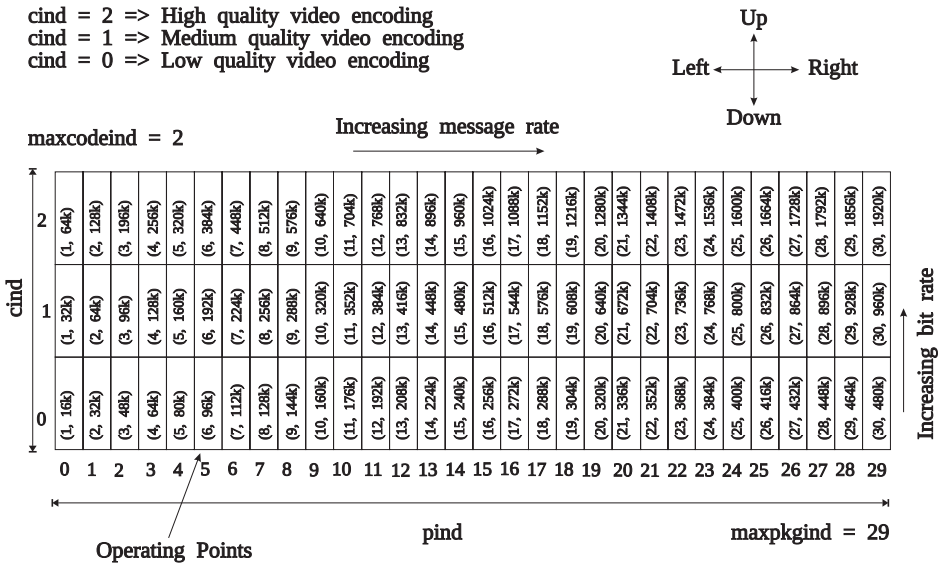


Figure 4-4: Conceptual View of Operating Point Array for Video in Figure 4-1

The primary advantage of viewing the operating points as an array is that implementing the actions associated with state transitions is easy. The transition actions in Figure 4-2 directly correspond to simple indexing operations within the array. Moving ***up*** or ***down*** corresponds to increasing or decreasing, respectively, the index value in **cind** by 1. Similarly, moving ***left*** or ***right*** corresponds to decreasing or increasing, respectively, the index value of **pind** by 1. A ***slide*** corresponds to decreasing the value of **pind**, perhaps by more than 1. In systems that do not support every available message rate for every potential coding scheme, the implementation of the transition actions is slightly more complicated since for a particular operating point, a transition action may result in changes in both the message and bit rate. In the scheme presented here movement is always via a change in a single index.

The **state** variable holds the current state for the stream. The **natural** variable holds the stream's natural state. The ramp variables for each state (variables **rramp**, **wramp**, and **pramp** for the *retreat*, *wait*, and *probe* state classes, respectively) are used to count the number of consecutive feedback intervals in which the algorithm remains in the state class. The associated threshold variables hold the number of consecutive intervals that must pass in a state before a transition is enabled.

Each time the conference sender receives a feedback message from the conference destination, the conferencing application calls the routine **feedback_handler** (see Listing 4-2) to evaluate the state of the conference and potentially change the effective operating point of each stream. The conferencing application passes the feedback handler a C structure with the values received from the most recent feedback message from the conference partner. The feedback handler invokes the RS algorithm for audio and video streams individually. The **recent_success** routine selects a candidate operating point for each stream independently using the existing state of the stream and the feedback information associated with the particular stream. We discuss the actual algorithm in more detail later in this chapter. The **feedback_handler** then determines whether the delivered video frame rate is low enough to trigger the "poor video" heuristic. If so, the new audio stream operating point is set to the operating point in (t) with the lowest message rate. This operating point is identified by the **min_inEST** routine (Listing 4-2). Finally, the **feedback_handler** calls the **set_stream_OP** once for each of the audio and video streams to set the effective operating points for the next feedback interval. These operating points remain in effect until the next feedback message arrives and the feedback handler is invoked again.

```
/* "poor" video fps */
#define POORVIDEORATE 20

/*----------------------------------*/
/* min_inEST                        */
/*   Pick min pkg index still in EST */
/*   or min of all possible).       */
/*----------------------------------*/
void min_inEST(int fb_IT,
               int latency,
               stream_defn *s)
{
   int i, newpind;

   s->state = wa;
   s->wramp = 0;

   /* get index of largest point */
   /*   satisfying feedback msg  */
   /*   interarrival time (fb_IT)*/
   newpind = getpkgindex(fb_IT, s);

   /* find smallest point that */
   /*   satisfies fb_IT        */
   /*   and induced latency    */
   for (i = newpind;
      ((i >= 0) &&
   (s->op[i][0].maxlatency >= latency));
       i--);

   /* set min feasible msg rate */
   if ((i + 1) <= s->pind)
      s->pind = i + 1;
} /* min_inEST */
```

```
/*----------------------------------*/
/* feedback_handler                 */
/*   Called for each feedback msg.  */
/*   Feedback values passed via     */
/*   feedback_header structure.     */
/*----------------------------------*/
void feedback_handler(feedback_header *FB)
{
   /*-------------------------*/
   /* select new OP for video */
   /*-------------------------*/
   recent_success(FB->video_interarrival,
      FB->video_latency -
      FB->min_video_latency,
      &video_op,
      video_op.ITtolerance,
      video_op.LATtolerance);

   /*-------------------------*/
   /* select new OP for audio  */
   /*-------------------------*/
   recent_success(FB->audio_interarrival,
      FB->audio_latency -
      FB->min_audio_latency,
      &audio_op,
      audio_op.ITtolerance,
      audio_op.LATtolerance);

   /*---------------------------------*/
   /* Poor Video Heuristic:           */
   /*   If new video rate too low, use */
   /*   lowest feasible audio msg rate.*/
   /*---------------------------------*/
   if (video_op.op[video_op.pind].msgrate
      < POORVIDEORATE) {
      min_inEST(FB->audio_interarrival,
               FB->audio_latency -
               FB->min_audio_latency,
               &audio_op);
  } /* endif */

  /*---------------------------------*/
  /* Tell conferencing system new OPs */
  /* to use for each media stream.    */
  /*---------------------------------*/
  set_stream_OP("audio", &audio_op);
  set_stream_OP("video", &video_op);
}
```

Listing 4-2: **`feedback_handler`** Routine

Listing 4-3a and Listing 4-3b show the implementation of the **recent_success** routine. Listing 4-3a shows the portions of the **recent_success** routine that test whether or not the current operating point is in $EST_S(t)$ and the portions that implement the failure transitions. Listing 4-3b shows the success transitions. The implementations of the state transition actions (*e.g.*, ***up***, ***left***, *etc*.) used in **recent_success** are given in Listing 4-4. Systems that do not support each coding scheme for each possible message rate may have a slightly more complicated implementation of the transition action routines. However, the base routine (*i.e.*, Listing 4-3) remains the same.

The feedback handler passes the **recent_success** algorithm the message interarrival time (called **fb_IT**) for the associated stream, the estimated stream network latency, a pointer to the stream definition, the interarrival time tolerance, and the latency tolerance. The algorithm first determines the highest message rate index that has a message period less than or equal to the measured message interarrival time. This is motivated by the message interarrival heuristic that says the message period should roughly match the delivered interarrival time because of the pacing introduced by the bottleneck server (this is the same heuristic used to compute $EST_S(t)$ using $IT_S(t)$ in Chapter 3). The index chosen by the interarrival heuristic is stored in **newpind**.

Next, the algorithm determines if the current operating point is a member of $EST_S(t)$ by determining if (1) the current message period is greater than or equal to the interarrival time of the messages at the destination, (2) the induced latency caused by this message rate does not exceed the budget imposed by fidelity constraints (*i.e.*, is the current operating point a member of $POP_S(t)$), and (3) the stream network latency is steady or decreasing (within a blurring threshold) when compared with the latency during the previous feedback interval (*i.e.*, the "latency" heuristic is invoked). If all these criteria are satisfied, then the current operating point is a member of $EST_S(t)$; otherwise, the current operating point is not in $EST_S(t)$.

If the current operating point is in $EST_S(t)$, the success transitions in Figure 4-2 are followed. Several successive successes eventually cause the stream to enter its natural state (from **wa**, **pa**, **wc**, or **pc**) when the maximum message and bit rates are finally achieved. If the current operating point is not in $EST_S(t)$, the failure transitions in Figure 4-2 are followed. If a ***slide*** is required, the message rate index selected by the interarrival heuristic (*i.e.*, **newpind**) is used as the new message rate index;

otherwise, transitions are always simple decrements of the message or coding scheme indices. If either of the indices reach 0, then no more adaptations can be made along the corresponding axis and the algorithm switches the characterization of the network congestion constraint. That is, when the message rate index reaches 0, the network is classified as capacity constrained and when the coding index reaches 0, the network is classified as access constrained. Thus, when all possible settings along one axis are exhausted, the algorithm switches to the other axis for further adaptation.

```
/*-------------------------------------*/
/* recent_success                      */
/*                                     */
/* Pick new OP for a stream based on   */
/* the available OPs, the current state,*/
/* most recent msg interarrival time   */
/* from feedback (i.e., "fb_IT"), and  */
/* and network latency from feedback.  */
/* (i.e., "latency").                  */
/*-------------------------------------*/
void recent_success(int fb_IT,
                    int latency,
                    stream_defn *s,
                    int ITtolerance,
                    int latencytolerance)
{
   int newpind;
   if (latency < 0) latency = 0;
   if (s->prevlatency < 0)
      s->prevlatency = latency;

   /*---------------------------------*/
   /* Message Interarrival Heuristic:  */
   /*  if fb_IT > period then slow link*/
   /*  if fb_IT < period then working  */
   /*     off queue                    */
   /*  if fb_IT = period then          */
   /*     residence time on slow link  */
   /*     <= period                    */
   /*---------------------------------*/
    newpind = getpkgindex(fb_IT, s);
    if ((s->pind - newpind)
         <= ITtolerance)
       newpind = s->pind;

   /*---------------------------------*/
   /* Is current OP a member of EST?  */
   /*  1.Does interarrival time match */
   /*    send period?                 */
   /*  2.Is overall latency acceptable*/
   /*    for this packaging? (POP)(t) */
   /*  3.Is latency trend reducing or */
   /*    increasing within tolerance? */
   /*    (Latency Heuristic)          */
   /*---------------------------------*/
   s->inEST = ((s->pind <= newpind) &&
     (s->op[s->pind][s->cind].maxlatency
         >= latency) &&
     ((s->prevlatency+latencytolerance)
         > latency));

   switch (s->inEST)    {
case false:  /* failure -> OP not in EST*/
{
   switch (s->state) {
      case ra:
         /* repeated failure */
         /* take a big slide */
         if (s->pind > 0)
            slide(s,
               newpind - s->rthreshold);
         else down(s);
            s->state = rc;
         break;

      case wa:
         slide(s, newpind);
         s->state = ra;
         break;

      case pa:
         if (++s->pramp > s->pthreshold) {
            left(s);
            s->wramp = 0;
            s->state = wc;
         } /* endif */
         break;

      case rc:
         /* repeated failure */
         /* take a big slide */
         if (s->cind > 0) down(s);
         else
            slide(s,
               newpind - s->rthreshold);
         s->state = ra;
         break;

      case wc:
         down(s);
         s->state = rc;
         break;

      case pc:
         if (++s->pramp >
               s->pthreshold) {
            down(s);
            s->wramp = 0;
            s->state = wa;
         } /* endif */
         break;
   } /* switch */
   break;
} /* false case */
```

Listing 4-3a: *Recent Success* - $EST_s(t)$ Membership and *Failure* Transitions

```
switch (s->inEST)
{
case true:  /* success -> OP in EST */
{
   switch (s->state) {
      case ra:
         s->wramp = 0;
         s->state = wa;
         break;

      case wa:
         if (++s->wramp > s->wthreshold) {
            if (s->pind < s->maxpkgind) {
               right(s);
               s->pramp = 0;
               s->state = pa;
            } else if (s->cind > 0) {
               s->state = wc;
            } else {
               s->state = s->natural;
            } /* endif */
         } /* endif */
         break;

      case pa:
         if (++s->pramp > s->pthreshold) {
            if (s->pind < s->maxpkgind) {
               right(s);
               s->pramp = 0;
               s->state = pa;
            } else if (s->cind > 0) {
               s->state = wc;
            } else {
               s->state = s->natural;
            } /* endif */
         } /* endif */
         break;

      case rc:
         s->wramp = 0;
         s->state = wc;
         break;

      case wc:
         if (++s->wramp > s->wthreshold){
            if (s->cind < s->maxcodeind){
               up(s);
               s->pramp = 0;
               s->state = pc;
            } else if (s->pind <
                        s->maxpkgind) {
               s->state = wa;
            } else {
               s->state = s->natural;
            } /* endif */
         } /* endif */
         break;

      case pc:
         if (++s->pramp > s->pthreshold){
            if (s->cind < s->maxcodeind){
               up(s);
               s->pramp = 0;
               s->state = pc;
            } else if (s->pind <
                        s->maxpkgind) {
               s->state = wa;
            } else {
               s->state = s->natural;
            } /* endif */
         } /* endif */
         break;
   } /* switch */
   break;
 } /* true case */
} /* switch */
s->prevlatency = latency;
} /* recent_success */
```

Listing 4-3b: *Recent Success - Success* Transitions

```
/*-----------------------------------*/
/* right                             */
/*                                   */
/* Increase to next avail stream     */
/*   msg rate.                       */
/* If at max msg rate, stay at max.  */
/*-----------------------------------*/
void right(stream_defn *s)
{
    s->pind = (s->pind < s->maxpkgind ?
               s->pind + 1:
               s->maxpkgind);
} /* right */

/*-----------------------------------*/
/* left                              */
/*                                   */
/* Decrease to next avail msg rate.  */
/* If at min msg rate, stay at min.  */
/*-----------------------------------*/
void left(stream_defn *s)
{
    s->pind = (s->pind > 0 ?
               s->pind - 1:
               0);
} /* left */

/*-----------------------------------*/
/* slide                             */
/*                                   */
/* Decrease to msg period at         */
/*    or below gap.                  */
/* If no period low enough,          */
/*    use min rate.                  */
/*-----------------------------------*/
void slide(stream_defn *s, int newpind)
{
    s->pind = (s->pind > newpind ?
               newpind:
               s->pind);
    if (s->pind < 0) s->pind = 0;
} /* slide */
```

```
/*-------------------------------------*/
/* up                                  */
/*                                     */
/* Increase to next avail bit rate for */
/*   current message rate.             */
/* If at max bit rate (encoding scheme), */
/*   stay at maximum.                  */
/*-------------------------------------*/
void up(stream_defn *s)
{
    s->cind = (s->cind < s->maxcodeind ?
               s->cind + 1:
               s->maxcodeind);
} /* up */

/*-------------------------------------*/
/* down                                */
/*                                     */
/* Decrease to next avail bit rate for */
/*   current message rate.             */
/* If at min bit rate (encoding scheme), */
/*   stay at minimum.                  */
/*-------------------------------------*/
void down(stream_defn *s)
{
    s->cind = (s->cind > 0 ?
               s->cind - 1:
               0);
} /* down */

/*-------------------------------------*/
/* getpkgindex                         */
/*                                     */
/* Find stream pkging index with period  */
/* less than interarrival time ("fb_IT").*/
/*-------------------------------------*/
int getpkgindex(int fb_IT, stream_defn *s)
{
    int i;

    for (i = s->maxpkgind;
         (i > 0) &&
         (s->op[i][0].msgperiod < fb_IT);
         i--);
    return(i);
} /* getpkgindex */
```

Listing 4-4: *Recent Success* Transition Actions Implementation

4.5 Conclusions

This chapter describes the *Recent Success* (RS) algorithm and gives one implementation of that algorithm. RS is essentially a search algorithm across the set of operating points. The search is directed by feedback from the conference partner and from the success of recent past changes to the stream. RS uses a set of heuristics to narrow the search space based on perceived current conditions. The algorithm implements the concepts and logic of the transmission control framework. As such, the algorithm can implement a spectrum of transmission control strategies ranging from pure scaling to pure packaging algorithms depending upon the available set of operating points.

Even though the transmission control framework is conceptually complicated, the resulting algorithm is very simple to implement and inexpensive to execute. For a single stream, the space required for the algorithm is $O(n)$ where n is the total number of operating points. In the worst case, the algorithm executes in $O(m)$ where m is the number of distinct message rates associated with the stream (due to the linear search in **`getpkgindex`**). Since most systems have only a relatively small number of operating points and few distinct message rates, the space and time requirements for the algorithm are usually very small. The typical execution path through the code executes only a few lines of code. Feedback messages are piggy-backed on media stream messages, so feedback does not introduce any new messages. The feedback data is small (in the current implementation, 12 bytes per stream per feedback message) and imposes a very small percentage overhead considering the size of the audio and video data.

The following chapters demonstrate that despite the simplicity of the algorithm, the algorithm's foundation in the transmission control framework makes it very successful at adapting to diverse network conditions regardless of the type of network congestion or the topology of the network itself. The algorithm described in this chapter is used throughout the remaining chapters to implement a variety of transmission control schemes. However, in all cases, the base algorithm remains the same. The operating points made available to the algorithm in each individual experiment constrain the degree of freedom available for adaptation.

Chapter V
Controlled Network Experiments

This chapter describes a set of controlled network experiments comparing the performance of several transmission control schemes on networks with access constraints, capacity constraints, and combination constraints. In this chapter, the access and capacity constraints are generated using synthetic traffic generators or modified routing code. All traffic on the networks is either part of the video conference media streams or specifically generated to introduce constraints. In other words, there is no "real" traffic on the networks and the test networks are isolated from all production networks. Chapter 6 discusses the results obtained using the transmission control framework on a production network.

The primary reason for using test networks rather than production networks is to insure the experiments are reproducible. Without controlling all traffic sources on a network, it is difficult to run experiments with similar traffic loads. We can also create specific types of constraints on test networks that we may or may not be able to create on a particular production network. We built the test networks used for the experiments in this chapter using "off-the-shelf" network components and computers. We prefer this approach to building a network simulator, since it avoids the potential problem of validating that the simulator accurately reproduces the network environment (*e.g.*, that the simulator accurately reflects the behavior of an Ethernet network).

The main problem with controlled network experiments is in determining the appropriate characteristics of the artificial traffic loads. There are no widely accepted traffic models for computer networks [71], so it is debatable if any controlled test or simulation accurately reflects true network traffic patterns. This chapter uses a plausible, although by no means exhaustive, set of traffic patterns produced with synthetic traffic generators. We do not present the synthetic traffic loads as a set of benchmark traffic patterns that mimic the behavior of production networks. Instead, we use the traffic patterns to create a specific set of network conditions from which

we draw some conclusions about the relative effectiveness of the transmission control algorithms.

This chapter is organized as follows. First, we give a brief description of the transmission control algorithms used in this chapter. Next, we describe our test environment including a description of the conferencing system used in the experiments and the test network. We then describe the graphs and tables used to report the quality of the conferences in each experiment. Our first set of experiments are over access constrained networks. The second set of experiments measure the results of video conferences carried over capacity constrained networks. Finally, we measure the results for a set of conferences carried over a network that is both access and capacity constrained. We measure the results for several transmission control algorithms and several network topologies for each type of congestion constraint. At the end of the chapter, we evaluate the relative success of each of the transmission control schemes in producing quality conferences and discuss some of our conclusions.

5.1. Basic Algorithm Descriptions

Unless otherwise noted, all experiments in this chapter use the same video conferencing system. Figure 5-1 shows the potential operating points for the conferencing system. The realization of the operating points depends on the network topology and is described as needed in the tests below.

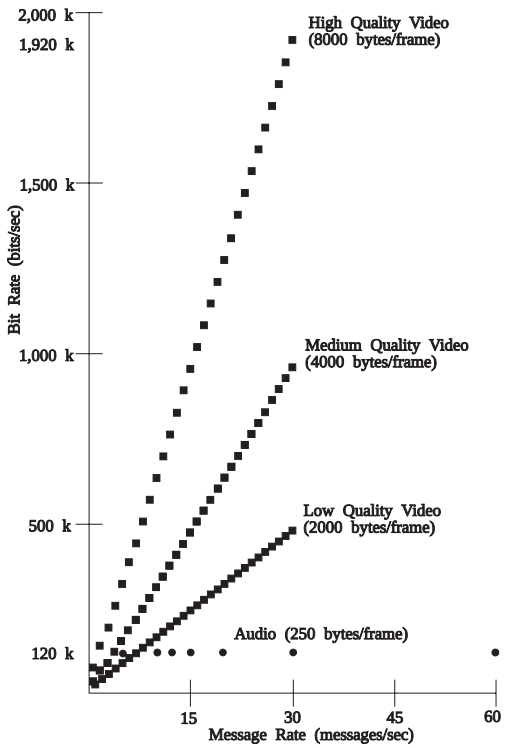


Figure 5-1: Potential Operating Points

All experiments use one of four transmission control policies. The simplest transmission control policy is the Baseline (BL) algorithm. This algorithm does not adapt to network congestion. BL uses a single operating point for each of the audio and video streams ((60, 120k) and (30, 1920k) for audio and video, respectively; see Figure 5-1). BL is a non-adaptive policy in that the video conferencing application always transmits at the highest message rate and bit rate available for each stream. Since there is no adaptation, BL is very easy to implement. BL transmits data as soon as it becomes available from the codecs and BL sends each media frame in a separate message. In the absence of network congestion, this strategy produces conferences with the highest fidelity and lowest induced latencies possible given the particular conferencing system; however, in a congested network this strategy often delivers inferior quality conferences when compared to those delivered by other transmission control schemes.

The Video Scaling Only (VSO) transmission control policy estimates network congestion and adapts the video coding scheme using a spatial video scaling technique. VSO chooses from the three potential video coding schemes corresponding to the operating points (30, 1920k), (30, 860k), and (30, 430k) in Figure 5-1. The high

quality video encoding, (30, 1920k), generates approximately 8,000 bytes per video frame. The medium quality encoding, (30, 860k), generates about 4,000 bytes per video frame. The low quality encoding, (30, 430k), generates about 2,000 bytes per frame. VSO does not use temporal scaling, so the video frame rate does not change (*i.e.*, VSO always transmits 30 video frames per second). VSO transmits each video frame as a separate message, so the video message rate never changes. VSO uses only a single operating point for the audio stream, (60, 120k). The basic idea behind VSO is to use video encodings with low aggregate bit rates when congestion is high and encodings with higher bit rates (and correspondingly higher fidelity) when congestion is low.

The third transmission control policy is the Temporal Scaling Only (TSO) algorithm. Like VSO, TSO is a video scaling algorithm, but TSO uses temporal rather than spatial scaling. Temporal scaling is a common transmission control scheme in video conferencing systems (*e.g.*, in H.261 [73]). In fact, temporal scaling is perhaps the most common scaling technique primarily because it is extremely easy to implement the algorithm. When congestion is high, the conferencing application on the source simply ignores some of the generated video frames. The ignored frames are never transmitted to the receiver. In our system, TSO only uses a single audio operating point, (60, 120k), but may use any high quality video operating point (*i.e.*, any operating point along the line from (1, 64k) to (30, 1920k)). TSO reduces the video bit rate (and indirectly the message rate) when congestion is high by lowering the effective frame rate of the high quality video encoding. When congestion is low, TSO increases the bit rate by raising the video frame rate.

The fourth transmission control policy is the Recent Success (RS) algorithm. The algorithm is based on the transmission control framework and may manipulate the message and bit rates of both the audio and video streams. We discuss the algorithm in detail in Chapter 4. For the experiments in this section, RS may use any of the operating points in Figure 5-1. RS is adaptive and changes the operating points of the media streams based on the perceived level of network congestion. For example, when RS detects high network congestion, the algorithm may respond by lowering the message rate or bit rate of one or both of the media streams based on a heuristic classification of the network (see Chapter 4). Similarly, when congestion decreases, RS may raise the message or bit rates of the media streams to improve the delivered quality of the conference. Note that VSO and TSO use the same algorithm as RS to

detect and measure network congestion (BL does not adapt, so does it not measure network congestion). VSO, TSO, and RS also use the same feedback interval and the same feedback data to make their adaptations (see Chapter 4). VSO, TSO, and RS differ only in their reaction to congestion.

5.2. Invariant Test Environment Features

This section describes aspects of the experiments that are constant for every experiment in this chapter, regardless of the degree or type of constraint present or the network topology.

5.2.1. Video Conferencing System

The video conferencing system used for the experiments in this dissertation was built at the University of North Carolina at Chapel Hill by the Distributed Real-Time Systems (DiRT) research group [103] [57] [61]. The hardware for this system consists of two Intel 80486-based (66 MHz) personal computers and associated video and audio hardware. The computers have 8 megabytes of RAM and use the IBM Microchannel (MCA) bus. The network adapters are IBM 16/4 MCA token ring adapters. The audio/video codec is the IBM/Intel ActionMedia I adapter [41]. Video is captured from a standard color video camera with RGB outputs. Audio is captured from a variety of sources including microphones and compact disc players. The test system is a one-way video conference; one computer captures audio and video and transmits it to a receiving machine, that plays the received data. The conferencing software on the receiving machine periodically provides feedback to the sending machine as described in Chapter 4.

The video conference application executes under a custom real-time operating system called YARTOS [57]. The conferencing application transmits all media and control data using the UDP/IP protocol [20]. The UDP/IP software layer was custom built for the YARTOS operating system. YARTOS provides strong guarantees on the scheduling of tasks within the operating system [59] and as a result the conferencing application has low and predictable latencies with very little jitter [103] when run over an unloaded network. Unfortunately, although jitter is controlled within the personal computers, congestion on the network may introduce jitter and loss into the media streams. Over a single hop network, the conference sender can directly measure the impact of network congestion (*e.g.*, by directly measuring the time needed to transmit

data on the network) and adapt the packaging of frames into messages to match the level of network congestion [103]. However, this technique does not work over multiple hop networks because the source node cannot directly measure the impact of congestion on any hop except the first hop. This dissertation describes the addition of a transmission control policy to the experimental video conference system to deal with the problems of congestion on multiple hop networks.

5.2.2. Network Environment

We conducted all the experiments in this chapter on the test network shown in Figure 5-2. The data transmission rate for the token ring segments is 16 Mbits/second and for the Ethernet is 10 Mbits/second. The maximum transmission unit (MTU) on the token rings is 17,800 bytes unless otherwise stated. The maximum transmission unit of the Ethernet is 1,500 bytes. We can create different transmission paths between the video conference source and destination by changing the routing tables in routers *R1* and *R2* (see Figure 5-2). For experiments using only token ring segments, the source machine sends the audio and video data over its attached token ring segment to router *R1*. *R1* routes the data across the middle token segment to router *R2*. *R2* then routes the data across the third token ring segment to the destination machine. Feedback from the conference destination follows the reverse path back to the sender. Experiments using the Ethernet segment are similar, but data is transferred between *R1* and *R2* via the Ethernet segment rather than the middle token ring segment.

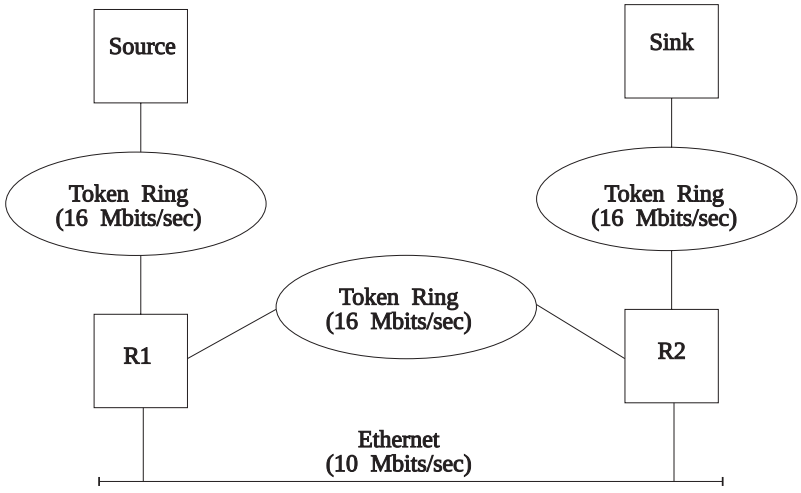


Figure 5-2: Experimental Network Configuration

The routers *R1* and *R2* are IBM RS/6000 model 20 computers running AIX version 3.25. We configured each of the network adapters (*i.e.*, the two token ring adapters and the Ethernet adapter in each router) with the default number of receive buffers in AIX (30 buffers). Unless otherwise stated, we configured the adapters to use a logical maximum transmission unit equal to the physical network maximum transmission unit. In the absence of network congestion, the routers are able to forward in real-time the full audio and video streams of all the conferencing systems evaluated.

5.2.3. Overview of Chapter 5 Experiments

Table 5-1 gives an overview of the experiments presented in this chapter. The table gives a very brief synopsis of each experiment. We discuss the specifics of each experiment and the results of the experiments throughout the rest of the chapter.

All the experiments are conducted over our experimental network (Figure 5-2). The experiments use one of three network configurations. Network configuration (A) uses the middle token ring as the path between the source and destination token rings. In this configuration, the MTU for all three token rings is set to 17,800 bytes. Network configuration (B) is the same as configuration (A), but the logical MTU for the middle token ring at router *R1* is set to 1,500 bytes. Network configuration (C) uses the Ethernet segment (see Figure 5-2) as the path between the source and destination token rings.

Each experiment has (1) an access constraint, (2) a capacity constraint, or (3) both an access and a capacity constraint. Each experiment measures the performance of several transmission control algorithms (BL, VSO, TSO, and RS). Some experiments use a high bit rate (HBR) conferencing system and some use a low bit rate (LBR) conferencing system. Some experiments compare the results for conferences experiencing different degrees of constraints (*e.g.*, moderately to severely constrained).

	Network Configuration		
Constraint Type	(A) Token Ring Backbone (MTU = 17,800 bytes)	(B) Token Ring Backbone (MTU = 1,500 bytes)	(C) Ethernet Backbone
(1) Access	BL, VSO, TSO, RS (HBR)	BL, VSO, TSO, RS (HBR)	(a) BL, VSO, RS (HBR) moderately constrained (b) BL, VSO, RS (LBR) moderately constrained (c) BL, VSO, RS (LBR) highly constrained
(2) Capacity	BL, VSO, TSO, RS (HBR)	BL, VSO, TSO, RS (HBR)	Results similar to B2; results not shown
(3) Both Access and Capacity	BL, VSO, TSO, RS (HBR)	Not evaluated	(a) BL, VSO, TSO, RS (HBR) moderately constrained (b) BL, VSO, TSO, RS (HBR) highly constrained (c) BL, VSO, TSO, RS (HBR) severely constrained

Table 5-1: Overview of Chapter 5 Experiments

5.3. Evaluating Test Results

The experiments in this and the following chapter use a set of graphs to illustrate the relative performance of each transmission control algorithm. These graphs are similar to those used to illustrate capacity and access constraints in Chapter 3, but include additional information about the actions taken by the control algorithm during the course of the conference. This section briefly describes each of the graphs presented with the experiments.

5.3.1. Frames per Second Graph

The frames per second (FPS) graph shows the number of audio and video frames delivered at the conference destination in each second of the conference. The *x*-axis of the graph is the seconds into the video conference. The *y*-axis is the number of frames delivered during the second. Video frame delivery is shown as a solid line and audio delivery as a dashed line (see Figure 5-5 (a) as an example). Some transmission control schemes may package more than one frame per message, so the number of frames delivered in a second does not necessarily match the number of messages delivered. Similarly, there is no direct relationship between the number of frames delivered in a second and the number of frames played in a second. The number of frames played is a function of both the frame delivery and the frame display policy (see [103] for a discussion of the effect display policies have on conference quality). The basic tradeoff with all display policies is to trade stream latency for smooth display. The FPS graph does not assume any particular display policy and does not measure the frames actually played. However, there is a relationship between the achievable performance of a display policy and the delivered frames. No display policy performs well when there are long periods with low frame delivery rates. All display policies benefit from predictable delivery rates and adaptive display policies that attempt to vary the buffering of frames depending on the jitter in the delivery stream are particularly effective when network jitter is dampened or controlled. All gaps reported here result from use of a simple, non-adaptive first-in-first-out display policy. The display policy never skips delivered frames and only buffers frames for display if more frames arrive during an interval than can be played. All transmission control algorithms use the same display policy.

The delivered frame rate measures the success of a transmission control policy from the perspective of the conference participant. The perceived fidelity of the video conference depends directly on the displayed frame rate and the displayed frame rate depends on the frame delivery rate. The conferencing systems used in the following experiments generate audio at 60 FPS and video at a maximum of 30 FPS, so ideally audio is delivered at 60 FPS and video at 30 FPS throughout the life of the conference. In practice, congestion may limit delivery of the generated rates. When evaluating the FPS graphs, conferences with high delivery rates are preferable to those with low delivery rates. If two conferences deliver approximately the same frame rate, we prefer the conference with the smaller variance in frame delivery. As discussed in

Chapter 1, we measure video fidelity primarily with the number of delivered frames per second, while with audio we focus on the number of gaps. We consider delivered video frame rates of 15 FPS or more to be acceptable and rates below 5 FPS to be non-interactive. We consider frame rates differing by 5 or more FPS perceptually distinguishable. In the system used for these experiments, audio samples have a duration of approximately 16 *ms*, so all audio gaps are perceptible and conferences with fewer audio gaps have better audio fidelity than conferences with more gaps. Since the perceived quality of the conference depends more on audio than video, we prefer conferences with high audio quality over those with low audio quality, even if the conference with lower audio quality has higher video quality.

5.3.2. Message Loss Graph

The message loss graph shows the number of messages lost during each second of a conference. The *x*-axis is seconds into the conference and the *y*-axis is the number of messages lost during that second. The number of messages includes both audio and video messages. Ideally, the message loss is zero throughout the life of the conference and conferences with lower message loss are preferable to those with higher loss. If messages are lost, then the video conference introduced messages into the network that were never delivered, wasting network resources and contributing to network congestion. As a "good network citizen," a transmission control policy should match transmissions to the level currently supported by the network and avoid sending messages with a low probability of delivery. Figure 5-5 (b) shows an example of a message loss graph.

5.3.3. Audio and Video Latency Graphs

Figure 5-5 (c) and (d) show examples of audio and video latency graphs. The *x*-axis is the number of seconds into the conference and the *y*-axis is the average latency in milliseconds of the messages delivered during that second. Conference latency directly affects the conference quality as perceived by the conference participants [51]. End-to-end latency, the latency between capture of the media frame at the source machine and display at the destination machine, is the best measure of conference latency in terms of perceived quality. However, the codec latency and the display policy at the destination machine may have a significant effect on the end-to-end latency. The emphasis here is on the latencies over which the transmission control

policy has some degree of control, namely the delivered, network, and induced latencies. The *delivered latency* is the difference between when a frame is available for transmission at the source machine and when it arrives at the network interface on the destination machine. The delivered latency is shown as solid lines on both the audio and video graphs.

The audio latency graph also shows the average induced latency over a second for the audio stream. This latency is shown as a dashed line. The *induced latency* is the difference between the time when the frame is available for transmission at the sending node and the time the frame actually begins transmission on the network. With many transmission control policies the induced latency is typically quite short, but with policies that sometimes transmit multiple audio frames per message (*e.g.*, the RS algorithm), the induced latency may sometimes be significant.

Network latency is the difference between delivered latency and induced latency. In the experiments reported here, the induced latency for video is always small since there is no video packaging. In this case, we use the delivered and network latencies interchangeably. The induced latency for the audio may be significant, so we distinguish among the three latencies when discussing the audio streams.

As discussed in Chapter 1, when comparing delivered latencies, we have adopted a guideline of 250 milliseconds for the maximum acceptable delivery latency for both streams. We consider all latencies below 250 *ms* acceptable and all above unacceptable. Although a fixed threshold is too simplistic for all continuous media applications in all environments, the guideline provides a useful measure for acceptable conference latency. We compare the latencies of two conferences by considering the number of times the conference streams exceed the 250 *ms* guideline and the duration of the violations. If two conferences have similar performance relative to the guideline maximum latency, we prefer the conference with the lower average latency. If two conferences have the same average latency, we prefer the conference with the more consistent latency (smallest standard deviation). We consider latency differences greater than 50 *ms* perceptually distinguishable and consider audio and video synchronized if the delivered latencies of the two streams are within 50 *ms* of each other.

5.3.4. Audio and Video Transmission Graphs

The audio and video transmission graphs (*e.g.,* see Figure 5-5 (e) and (f)) show the adaptation of the streams by the transmission control algorithm. The FPS, message loss, and latency graphs show the performance delivered by a transmission control scheme. The audio and video transmission graphs show how the transmission control scheme achieves its results. The audio transmission graph has two components. The line labeled "Audio Frames/Message" shows the average number audio frames transmitted per message over a second. The *x*-axis is the seconds into the conference and the *y*-axis is the number of frames per message (the conference in Figure 5-5 (e) always transmitted a single audio frame per message). Audio is always transmitted as either a stereo or a monaural stream. The lines labeled "Stereo" or "Mono" indicate whether the conference was broadcast in stereo or monaural audio over a particular period. The presence of a line under the "Stereo" label means that over the seconds covered by the line, the conference was using stereo audio. Similarly, the presence of a line under the "Mono" label indicates the conference was transmitted in monaural (the conference shown in the example in Figure 5-5 (e) was only transmitted in stereo). Stereo and monaural are boolean values and the lines associated with stereo and monaural do not correlate with the *y*-axis.

The video transmission graph shows how video frames where transmitted over the course of the conference. There are three possibilities. Lines under the labels "High Quality," "Medium Quality," and "Low Quality" indicate that for the duration of the line, the video stream was transmitted using high, medium, or low image quality, respectively (the conference shown in the example in Figure 5-5 (f) only transmitted high quality video). These lines are analogous to the stereo and monaural lines in the audio transmission graphs and give an indication of the fidelity of the transmitted stream. The line labeled "Video frames/sec" shows the average generated frame rate for each second of the conference. The *x*-axis is the seconds into the conference and the *y*-axis is the average number of video frames generated during the second. The maximum possible video frame rate for the systems described here is 30 frames per second. Some of the transmission algorithms (*i.e.,* TSO and RS) change the video frame rate in response to network congestion and these changes are reflected in the "Video frames/sec" line over the course of the conference. The combination of frame rate and coding scheme measures the *transmitted* video fidelity and perceptual quality, but it is the *delivered* video frames (*i.e.,* from the FPS graph) that more accurately

reflects the perceived quality of the video. Transmitting high quality images at high frame rates may not lead to the best delivery of video across a congested network.

5.4. Dynamic Access Constraints

This section describes a set of controlled network experiments measuring the video conference quality achieved by several transmission control strategies on access constrained networks. The experiments are controlled in the sense that all traffic other than the video conference traffic is generated using a set of traffic generators. Using traffic generators allows creation of "pure" access constraints, where, for example, the only constraint in the network is competition for access to a particular shared communications link. Since the traffic generators are programmed to follow a prescribed script, experiments can be reproduced.[1]

The purpose of the experiments in this section is to show that access constraints can significantly degrade the quality of a video conference, but that through judicious adjustment of the conference message rate, we can limit the effects of the access constraints and preserve conference quality. We evaluate several transmission control strategies in a number of network environments. The network environments vary in the severity of the access constraint, the network technology, and the network topology.

5.4.1. Dynamic Access Constraints on Token Rings

5.4.1.1. Generating Access Constraints

We created access constraints by attaching traffic generators to the middle token ring segment or to the Ethernet segment, depending upon the experiment. Traffic generators are programs running on Intel 80386/80486-based machines that individually produce traffic on the network following a script describing a desired traffic pattern. The script may contain multiple phases that are repeated until the traffic generator is manually stopped. Each phase is described by distributions for the size of generated packets, the inter-packet waits (the elapsed time between generation

[1]The traffic generator scripts are statistically based, so repeated experiments are not identical, but are statistically similar.

of successive packets), and the length of the phase. The generators support constant, uniform, normal, and exponential distributions.

Table 5-2 shows a sample script for a traffic generator. The sample script has 4 phases. The length of the first phase is calculated using a normal distribution with a mean of 5 seconds and standard deviation 0.5 seconds. The seconds between generation of successive packets (*i.e.*, the "wait" time) during the first phase is derived using a normal distribution with mean 0.5 seconds and standard deviation 0.25. The size in bytes of the packets is constant with 4,096 bytes per generated packet. The remaining phases are described similarly, but with different distributions and parameters for the interval, inter-packet wait, and packet size. When the last phase completes, the traffic generator returns to the first phase and repeats the cycle indefinitely. Each traffic generator may use a different traffic script.

The traffic generators create access constraints by generating competition for use of the shared medium network. We can control the length and severity of access constraints by changing the number of traffic generators and the phase distributions and parameters. On the token ring networks, the traffic generators access the network via the direct token ring interface [48]. On the Ethernet network, the traffic generators access the network via an NDIS interface [1].

```
intervals 4
size constant 4096
wait normal  0.5  0.25
interval normal 5.000 0.500
*
size normal 100 10
wait normal 0.5  0.25
interval normal 17.000 20.000
*
size constant 16000
wait constant 0.00001
interval normal 6.000 5.000
*
size normal 100 10
wait normal 0.5  0.25
interval normal 21.000 20.000
```

Table 5-2: Sample Traffic Generator Script

5.4.1.2. Test Environment

The experiments in this section use the Baseline (BL), Video Scaling Only (VSO), Temporal Scaling Only (TSO), and Recent Success (RS) transmission control schemes on the three hop token ring network shown in Figure 5-3. Traffic generators (*P1*, *P2*, *P3*, and *P4* in Figure 5-3) are attached to the middle token ring segment and cause an access constraint on that segment due to the competition for free tokens. This in turn causes router *R1* to become congested. The candidate transmission schemes attempt to alleviate the effects of this congestion.

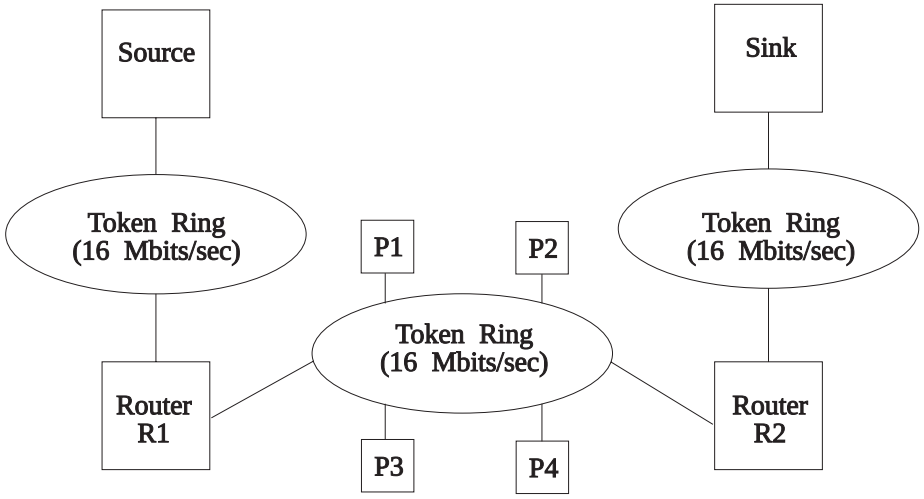


Figure 5-3: Three Hop Token Ring Network

Table 5-3 shows the traffic scripts for the four traffic generators. The effect of this combination of traffic generators and scripts is to generate a varying access constraint on the middle token ring network. At times, traffic on the middle network is low to moderate and imposes no access constraint on the network. At other times, the traffic is much higher and the access constraint correspondingly more severe. During the peak periods, the average time to acquire a free token, as measured by the IBM Trace and Performance (TAP) tool, is 24 milliseconds and network utilization reaches 99 percent. The four traffic generators can be thought of as either normal user nodes attached to the middle network or as bridges/routers feeding a backbone token ring from other spur networks. The latter configuration, with “floor” or “building” rings bridged or routed onto a shared “backbone” ring is a common LAN topology. The intent of this traffic pattern is not to mimic any particular traffic experienced on a

production LAN, but rather to illustrate the effects of a severe access constraint on the quality of a conference. It also shows that only a few active machines are needed to create significant congestion problems.

intervals 1 size constant 16000 wait constant 0.0001	intervals 3 size constant 100 wait constant 0.25 interval constant 1.00 * size constant 16000 wait constant .00001 interval constant 10.00 * size constant 10 wait constant 0.5 interval constant 2.00	intervals 5 size normal 100 10 wait normal .5 .25 interval normal 17.0 20.0 * size constant 4096 wait normal .5 .25 interval normal 5.000 0.500 * size constant 16000 wait constant .00001 interval normal 6.00 5.00 * size normal 100 10 wait normal .5 .25 interval normal 21.000 20.000 * size normal 100 10 wait normal .05 .05 interval normal 7.00 2.00	intervals 3 size normal 100 10 wait normal .5 .25 interval normal 35.0 13.0 * size constant 16000 wait constant .00001 interval normal 33.0 7.0 * size normal 100 10 wait normal .05 .05 interval normal 12.0 4.0

Table 5-3: Traffic Generator Scripts for Access-constrained Token Ring

5.4.1.3. Experimental Results

Figures 5-5, 5-6, 5-7, and 5-8 show the results from video conferences using the BL, VSO, TSO, and RS algorithms, respectively, on the network in Figure 5-3 with dynamic access constraints. Table 5-4 summarizes the results of the experiment. We can see from Figure 5-5 that a non-adaptive transmission strategy such as the BL algorithm produces poor quality conferences in an access constrained network. In this experiment, audio is generated at 60 frames per second, but there are several long periods when audio frame delivery is below 30 frames per second (Figure 5-5 (a)). During these periods audio is unintelligible. Video frame delivery also drops during the congested periods. Both audio and video experience extremely high latencies (about 800 *ms*) during the congested periods (parts (c) and (d)). Because router *R1* has difficulty obtaining sufficient tokens on the middle network during the congested periods, long queues build in *R1*, leading to the excessive latencies and eventually the high message losses shown in part (b). Since BL does not adapt the audio or video

message or bit rate at any time during the conference (the lines in parts (e) and (f) are flat lines), conference quality is dramatically affected by the congestion in the network.

Figure 5-6 shows the performance of a video conference using VSO in the access constrained network. VSO delivers better audio than BL. There are 1,883 16 *ms* audio gaps in the conference using VSO compared with 3,407 with BL (see Table 5-4).[2] However, although VSO improves audio performance, the audio quality is still poor. Although VSO lowers the video quality from "high" to "low" during periods of congestion (part (f)), the audio and video streams still have periods of extremely high latencies (700 to 800 *ms*; parts (c) and (d)) and experience significant loss (part (b)). During congested periods, VSO cuts the video bit rate to 25% the rate used during non-congested periods, but there is little effect on the delivered quality of the conference.

The reason for the small impact of the bit rate changes made by the VSO algorithm is that the experimental network is constrained by packet rate not by bit rate. In particular, *R1* is constrained by the time required to acquire a free token on the middle token ring segment. In Chapter 3, we discussed how we could calculate the packet rate resulting from a particular operating point for any hop in the network path. In particular, we can run the algorithm in Listing 3-1 for each operating point $(m_s, b_s) \in OP_s$ to calculate the associated packet rate for each hop. This association is represented with a triple, $(m_s, b_s, p_{s,k})$. We call the set of triples resulting from the operating points in OP_s the *realization* of the operating points at hop k along the path. Figure 5-4 shows the realization of the operating points in Figure 5-1 for a token ring segment. To make the figure easier to read, we have represented the realization of the operating points as columns rather than points in a three-dimensional space (*i.e.*, the point (x, y, z) is represented as a column from $(x, y, 0)$ to (x, y, z)).

The realization shows that for this network (Figure 5-3) changing only the video coding scheme has no effect on the aggregate packet rate of the conference (*i.e.*, all three coding schemes generate the same packet rate on all hops of the token ring for a

[2]We consider each frame time where there is nothing to play a "gap." We count each of these gaps independently, even if several consecutive frame times have nothing to play.

given frame rate).[3] This means it is impossible for transmission schemes that rely solely on coding scheme changes to effectively counteract the effects of access constraints. Since VSO does not manipulate the conference message rate, the algorithm cannot directly address the access constraint and there are no indirect changes to the conference packet rate due to the bit rate changes. Furthermore, our experiences with our AIX routers have led us to believe that the AIX TCP/IP software running in router *R1* allocates buffers on a packet basis rather than a byte basis. In this case, reducing the video bit rate not only fails to eliminate the access constraint on the middle token ring, but also has little effect on buffer consumption in *R1*.

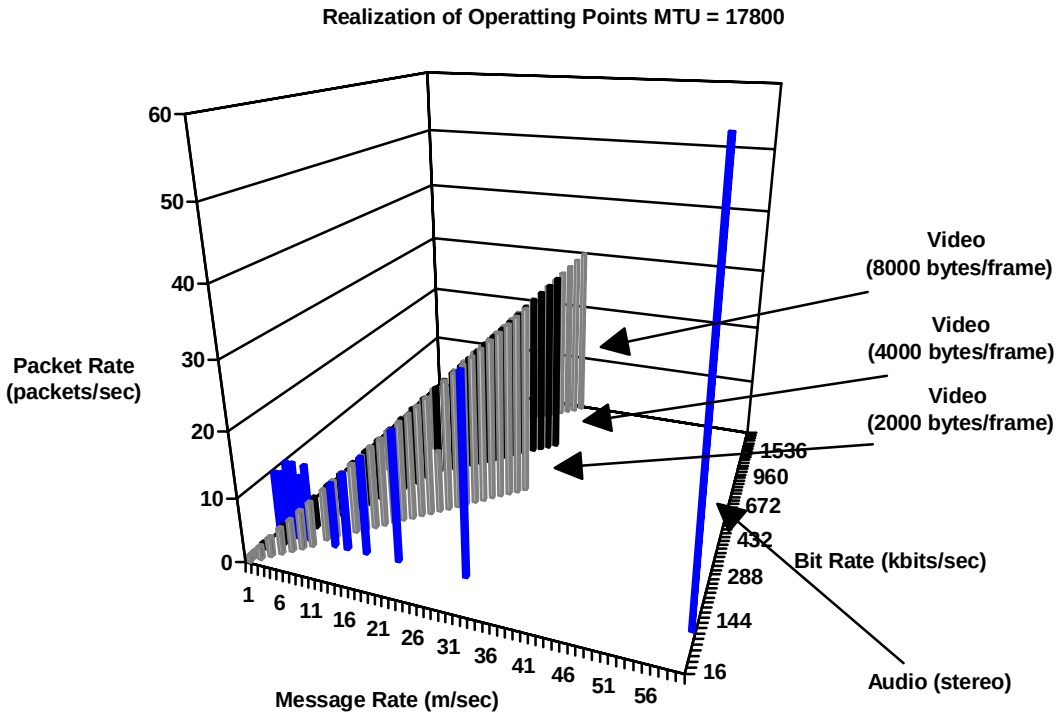


Figure 5-4: Realization of Operating Points on Token Ring Network

Unlike VSO, the TSO algorithm can directly address the conference message rate (and thus packet rate) by reducing the frame rate of the video stream. Unfortunately, Figure 5-7 shows the conference results are still poor. Audio delivery rates are

[3]Although changing the video coding scheme has no effect on the realized packet rate, it does, of course, affect the perceived quality of the video stream.

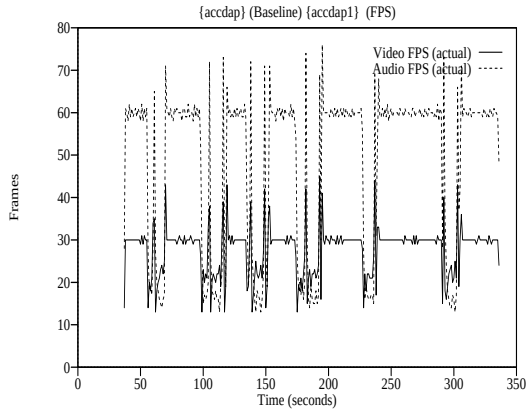
improved (part (a)) and fewer messages are lost (part (b)); however, the audio and video streams still have periods with extremely high latencies (700 to 800 *ms*; parts (c) and (d)) and video frame delivery often borders on non-interactive (part (a)). There are slightly fewer audio gaps under TSO (1,387 gaps versus 1,883 with VSO), because of the lower number of video packets transmitted across the network during the congested period. TSO transmits as few as 5 video packets per second during the congested periods (part (f)). For perceptual reasons, we constrained TSO to video frame rates of at least 5 frames per second (*i.e.*, we removed the video operating points with frame rates below 5 per second). During the congested periods, the algorithm reaches the limits of its possible adaptations. Unfortunately, that adaptation is not enough to counteract the congestion in the network, so even though the audio performance is better than with BL and VSO, TSO still delivers poor audio and the video performance is worse (the average delivered video FPS is 21.20 versus 27.81 with BL and 27.66 with VSO). The TSO algorithm directly addresses the access constraint, but the response is inadequate because TSO only manipulates the video message rate. Unfortunately, in this network, audio is the dominant source of packets, generating at least twice as many packets as video (see Figure 5-4).

Figure 5-8 shows the performance of a video conference using the RS transmission control algorithm. The quality of the delivered conference is much higher than with any of the three previous algorithms. The audio frame delivery rate is high throughout the conference (part (a)), even during congested periods (*e.g.*, between 100 and 150 seconds into the conference). Audio does not experience the periods of unintelligibility that plagued the first three conferences. There are only 51 audio gaps during the course of the conference. Video quality is good over the course of the conference, both in terms of delivered frame rate (part (a)) and coding quality (part (f)). Except at the onset of the congested periods when the algorithm is adapting to the network congestion, video is delivered at 30 frames per second using the high quality coding scheme. The long periods of high media stream latency seen with the other algorithms are reduced to very short intervals of high latency that occur when the algorithm is adapting to increasing congestion (parts (c) and (d)). Furthermore, there is almost no message loss (part (b)). The key to the success of this algorithm is the reduction in message rate resulting from the packaging of audio frames into messages. Part (e) shows the average number of audio frames carried by each message over the life of the conference. By changing the audio message rate, the RS algorithm is able to lower the number of times *R1* must acquire a token on the

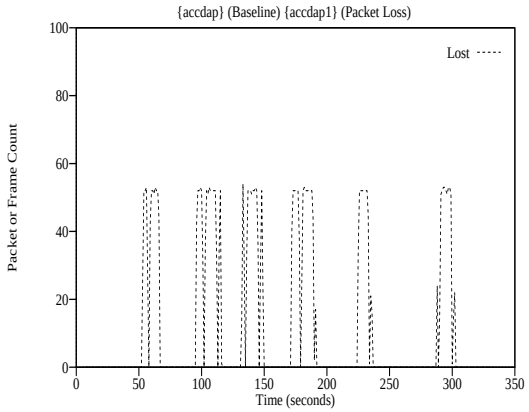
congested middle token ring segment. In particular, the number of tokens needed to carry audio data is reduced by as much as a factor of ten (from 60 tokens per second to 6; see Figure 5-4). This change in audio packaging coupled with short reductions in the video frame rate enable RS to quickly react to access constraints and lower the demands of the conference to a level that the network can support. The conference uses the maximum bit rate available to the conferencing system except during the relatively short periods when the video frame and bit rates are reduced at the onset of congestion. The access constraints in the network do not force extended periods of lower bit rate streams, but instead are addressed by reducing the message rates of the conference streams.

5.4.1.4. Conclusions about Access Constraints on Token Rings

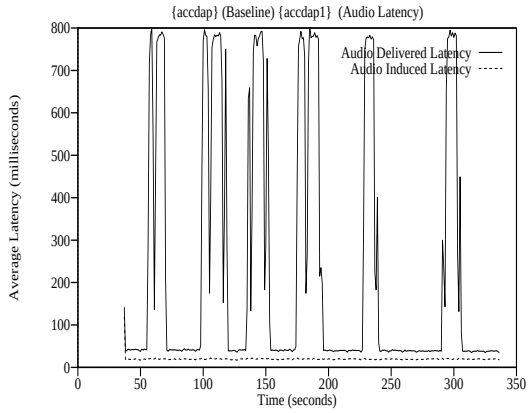
The major conclusion to draw from the access constraint experiments on 16 Mbits/second token rings is that in networks where the MTUs that are large in relation to the size of the media frames, we can ameliorate the effects of access constraints by changing the message rate of the conference. Access constraints can severely degrade the quality of a conference and non-adaptive algorithms are unlikely to deliver adequate conference quality during congested periods. Depending on the characteristics of the network path, lowering the conference bit rate alone may be totally ineffectual when dealing with access constraints. At best, transmission control techniques that rely solely on bit rate reductions (*e.g.*, VSO) are limited to secondary packaging effects when dealing with access constrained networks. At worst, in networks with large MTUs there may be no secondary effects and bit rate reductions are futile. Transmission control techniques such as TSO that directly manipulate the stream message rate, but limit the adaptation to a single media stream (*e.g.*, video) may be unable to adapt sufficiently to ameliorate the congestion and may unnecessarily penalize the adapted stream compared with other media streams. By exploiting the packaging flexibility available on token rings, RS delivers a high fidelity, low latency conference even when there are severe access constraints and RS is far superior to the other transmission techniques in limiting conference message loss.



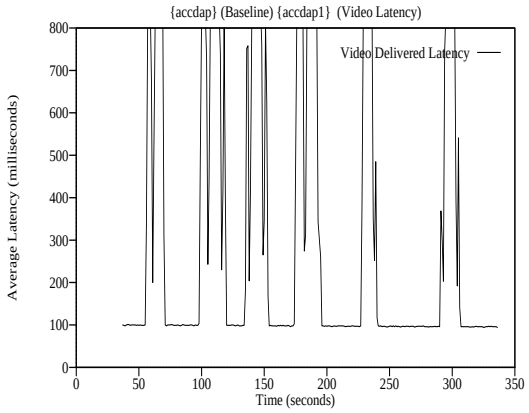
(a) Frames Per Second (FPS)



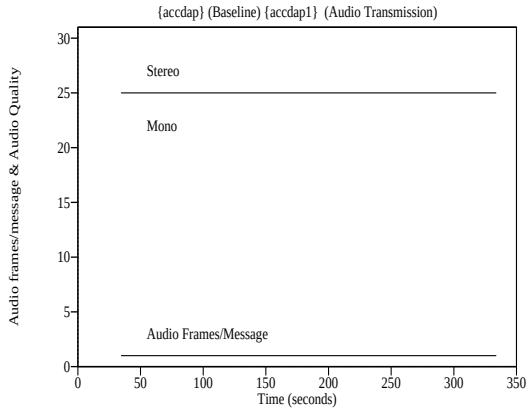
(b) Message Loss



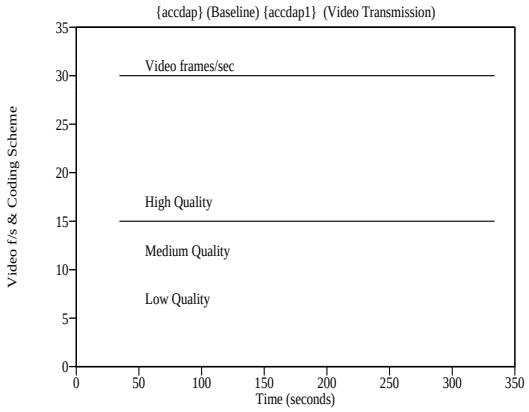
(c) Audio Latency



(d) Video Latency

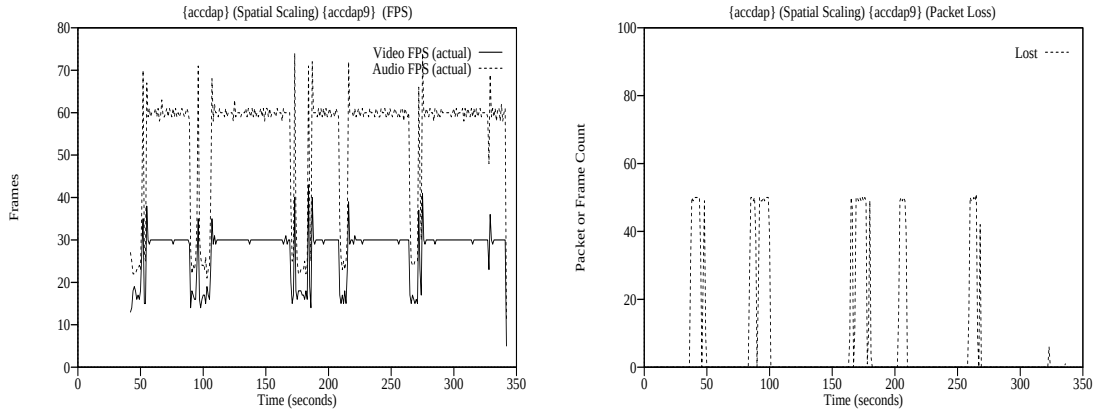


(e) Audio Transmission



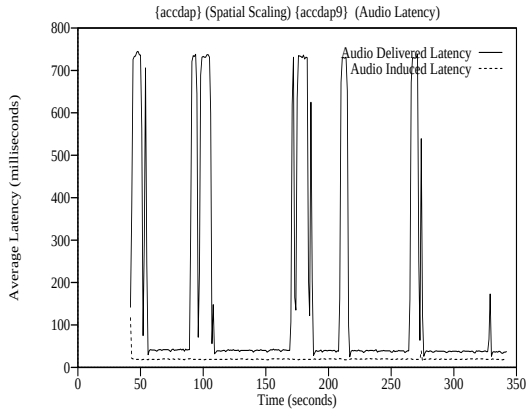
(f) Video Transmission

Figure 5-5: Access Constraint (TR-TR-TR) Experiment - Baseline

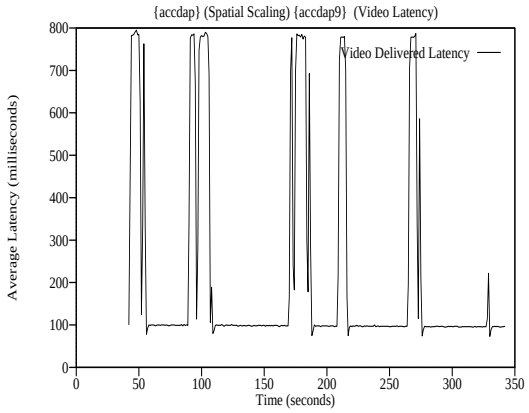


(a) Frames Per Second (FPS)

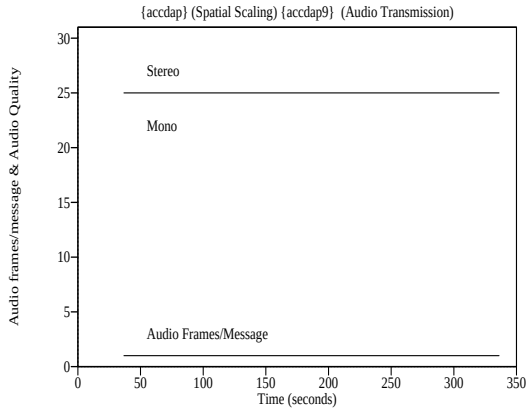
(b) Message Loss



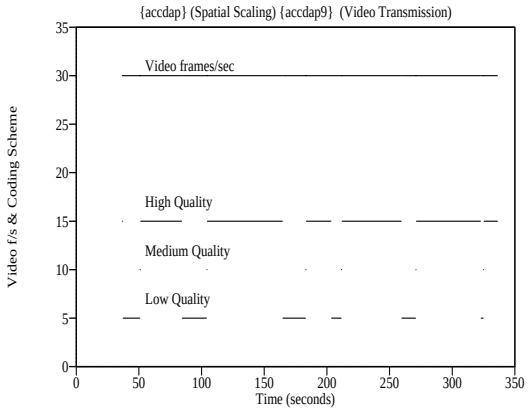
(c) Audio Latency



(d) Video Latency

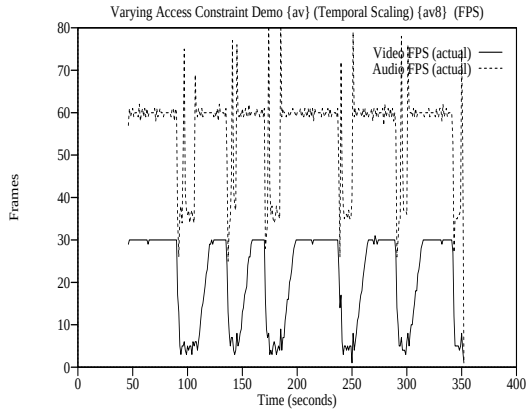


(e) Audio Transmission

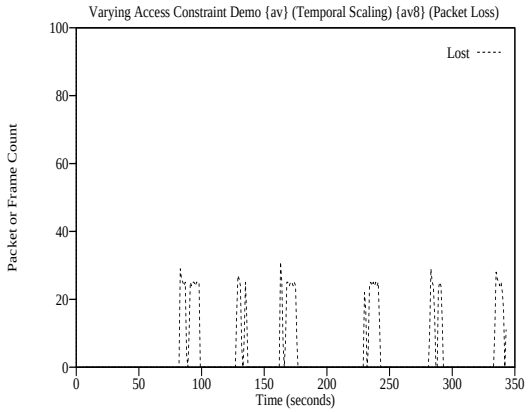


(f) Video Transmission

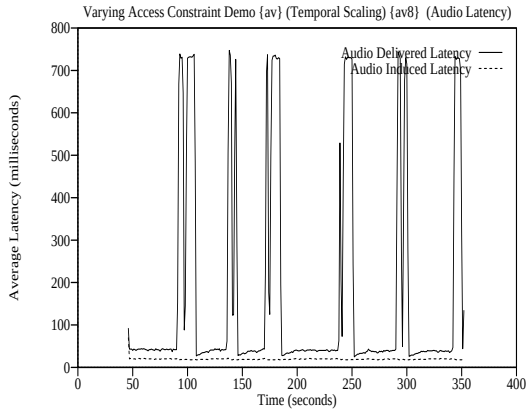
Figure 5-6: Access Constraint (TR-TR-TR) Experiment - Video Scaling Only



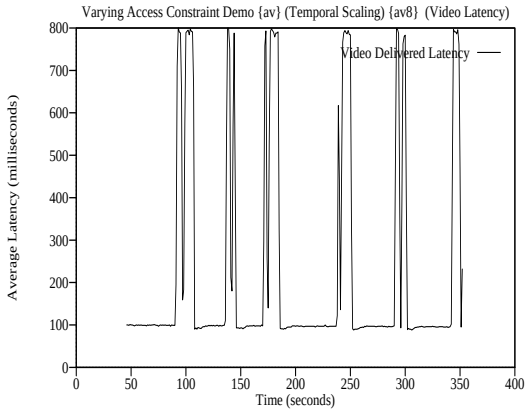
(a) Frames Per Second (FPS)



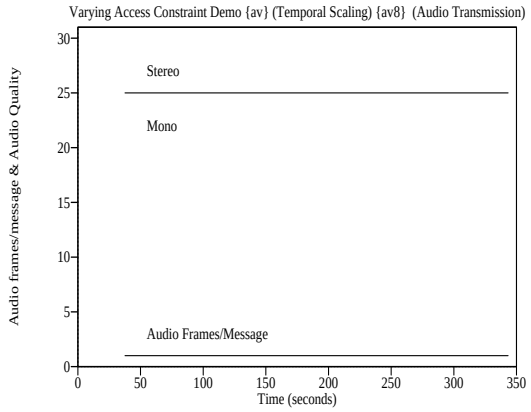
(b) Message Loss



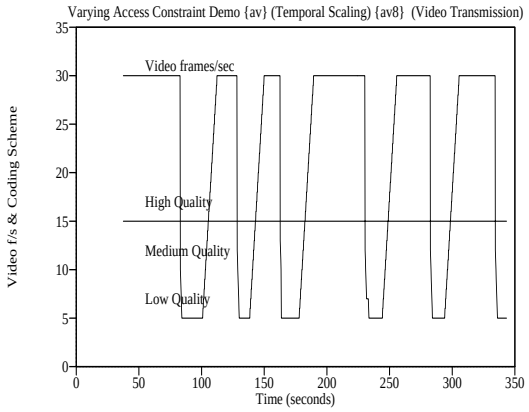
(c) Audio Latency



(d) Video Latency

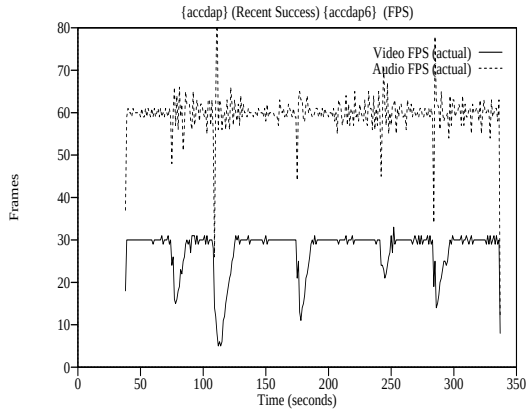


(e) Audio Transmission

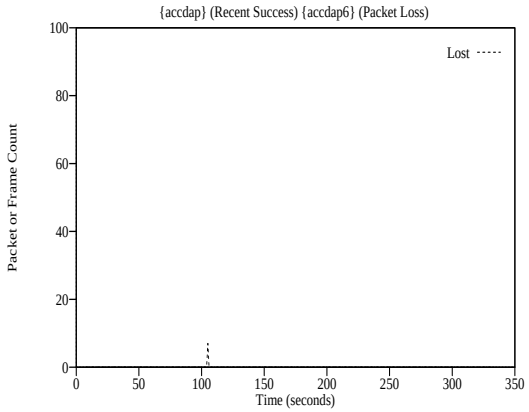


(f) Video Transmission

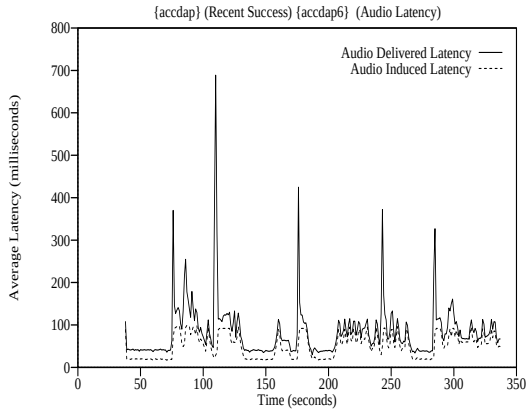
Figure 5-7: Access Constraint (TR-TR-TR) Experiment - Temporal Scaling Only



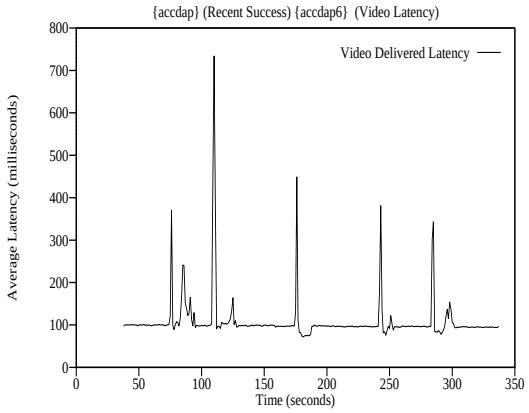
(a) Frames Per Second (FPS)



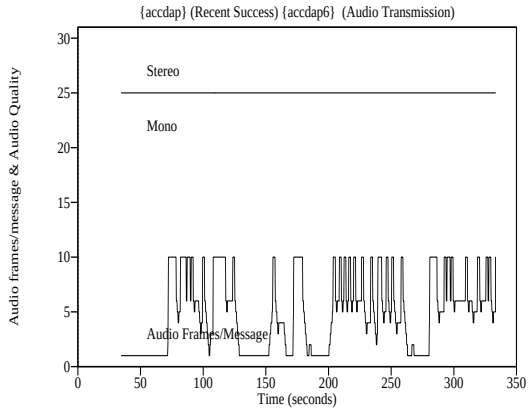
(b) Message Loss



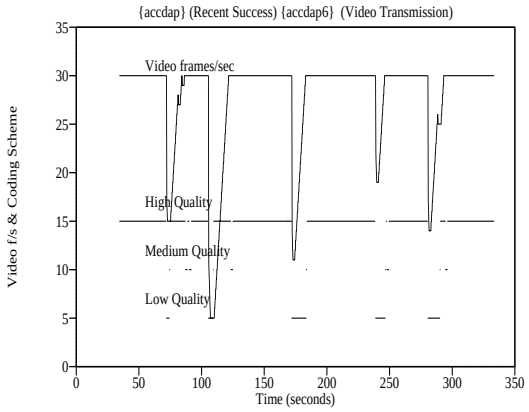
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-8: Access Constraint (TR-TR-TR) Experiment - Recent Success

Varying Access	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	48.61	53.62	55.33	59.76
Standard deviation	19.04	14.02	11.20	5.09
Minimum	13	11	1	12
Maximum	76	74	84	91
Mode/Median	60/59	60/60	60/60	60/60
Gaps	3407	1883	1387	51
Audio Latency: Delivered (Induced)				
Mean	234.41 (19.57)	155.66 (19.38)	163.68 (19.29)	83.81 (50.75)
Standard deviation	304.11 (0.83)	245.83 (1.34)	250.60 (0.96)	65.79 (28.08)
Minimum	35 (17)	25 (14)	25 (17)	30 (17)
Maximum	810 (22)	745 (39)	747 (22)	689 (99)
Mode/Median	39/41 (20/20)	40/40 (19/19)	40/40 (20/19)	40/69 (19/51)
Intervals > 250 ms	87	52	61	9
Audio Messages				
Frames Sent	17962	17987	18362	17936
Msgs(frames) Lost	3375 (3375)	1848 (1848)	1354 (1354)	2 (3)
Mean Frames Lost	11.25	6.14	4.41	0.01
Max Frames Lost	48	39	31	3
Video FPS				
Mean	27.81	27.66	21.20	27.76
Standard deviation	5.65	5.71	10.85	5.21
Minimum	13	5	1	5
Maximum	45	43	31	33
Mode/Median	30/30	30/30	30/30	30/30
Gaps	665	688	2667	650
Video Latency: Delivered				
Mean	296.07	212.73	223.79	108.92
Standard deviation	307.77	243.42	252.16	59.13
Minimum	94	73	88	72
Maximum	877	795	813	734
Mode/Median	96/97	97/98	97/98	97/97
Intervals > 250 ms	102	57	66	8
Video Messages				
Frames Sent	8980	8994	6563	8335
Frames Lost	640	670	40	5
Mean Frames Lost	2.13	2.23	0.13	0.02
Max Frames Lost	14	15	5	5

Table 5-4: Varying Access Constraint Experiment Summary

5.4.2. Access Constraints on a Token Ring with Limited MTU

5.4.2.1. Test Environment

The experiments in this section are very similar to those in the previous section. The network topology and traffic load are the same as the previous experiment. The only change is that on router *R1* (see Figure 5-3), we set the logical maximum transmission unit of the middle token ring to 1,500 bytes rather than 17,800 bytes. The intent of this experiment is to demonstrate, by changing a single variable from the previous experiment, how small network MTUs exacerbate access constraints. The experiment also has practical considerations because the default logical MTU used by many routers for token ring networks is the same as for Ethernet (*i.e.*, 1,500 bytes). Routers using the default MTUs are often found in production networks. In the previous experiment, each video frame is carried by a single token ring packet across the entire network (see Figure 5-4). In this experiment, video frames are too big to fit in a single packet on the middle network and must be fragmented (typically 1 video frame is fragmented into 6 packets). This greatly increases *R1*'s demand for tokens on the middle token ring. This results in a drop in conference performance. We show that the RS algorithm can still ameliorate the effects of the congestion, even in the fragmented environment, while the other transmission schemes are less successful.

5.4.2.2. Experimental Results

Figures 5-10, 5-11, 5-12, and 5-13 show the results from video conferences using the BL, VSO, TSO, and RS algorithms, respectively, on the network in Figure 5-3 with dynamic access constraints and the MTU on the middle token ring is set to 1,500 bytes. Table 5-5 summarizes the results. These experiments use the same traffic pattern as the previous set of experiments (Figures 5-5 through 5-8), but because of the logical MTU on the middle token ring, the demand for tokens has increased since it takes more packets to carry the same bit rate. Figure 5-9 shows the realization of the operating points from Figure 5-1 on the middle token ring segment in Figure 5-2.

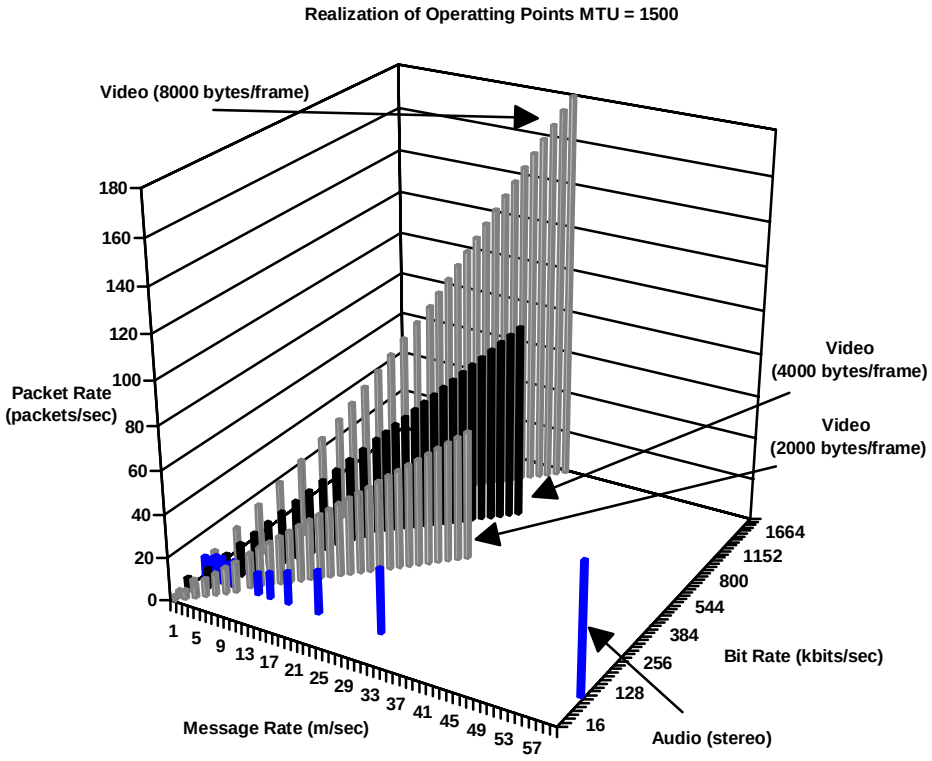


Figure 5-9: Realization of Operating Points on a Network with MTU=1,500

Figure 5-10 shows that the increased need for tokens results in significantly lower conference quality when using the BL algorithm than in the experiment with a larger MTU (Figure 5-5). The delivered frame rates for both the audio and video are very poor, with unintelligible audio and non-interactive video (part (a)). When the MTU is 17,800 bytes, the conference using BL had 3,407 audio gaps and 665 video gaps. This is poor performance, but when the MTU is 1,500 bytes, the conference performance using BL is much worse. In this case, there are 8,544 audio gaps and 8,311 video gaps (Table 5-5). Message loss is high throughout the experiment (part (b)). The audio stream is often delivered with high latency (over 700 *ms*; part (c)). The delivered video stream does not experience excessively high latency, but primarily because many of the video frames are lost within the network.

The VSO algorithm (Figure 5-11) performs better than BL, but the results are still poor. Audio fidelity is better than with BL, but is never good and is frequently unintelligible (part (a)). There are 4,419 audio gaps during the conference (Table 5-5). Video fidelity is also better than with BL, but there are several periods where the

delivered video frame rate is less than 5 frames per second and there are 4,044 video gaps during the conference. Many messages are lost (part (b)) and both the audio and video streams experience periods of high latency (parts (c) and (d)).

The TSO algorithm (Figure 5-12) does better than the either BL or VSO. Audio is often good, although there are still long periods of poor audio fidelity (*e.g.*, see part (a) between 250 to 300 seconds into the conference). The delivered video frame rate is more consistent than with VSO, although it never reaches the best levels achieved by VSO. This result is not too surprising, since VSO always sends 30 frames per second, adapting only the video coding scheme, and TSO changes the frame rate, but does not change the coding scheme. Unfortunately, the reduction in packet rates resulting from the lower video frame rate is insufficient to offset the access constraint. Message loss is high (part (b)) and both streams experience periods of high latency (parts (c) and (d)). There are 3,382 audio gaps and 6,341 video gaps during the conference. Nevertheless, compared with BL and VSO, TSO comes closer to achieving the same results on this experiment, where the middle segment has an MTU of 1,500 bytes, and the previous, where the MTU is 17,800 bytes. The reason for this is that where BL does nothing to affect the packet rate and VSO only achieves a secondary effect reduction of the packet rate, TSO directly adapts the packet rate by manipulation of the video message rate.

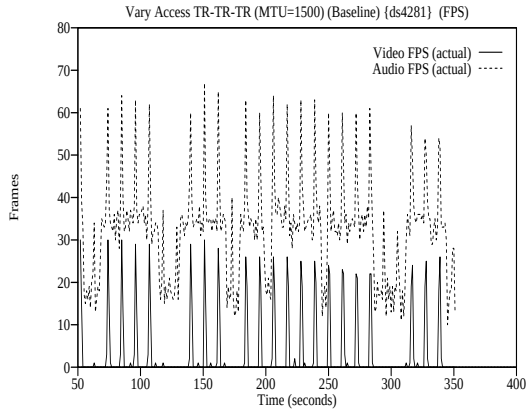
Figure 5-13 shows the results of the experiment using the Recent Success algorithm. RS gives significantly better audio display than any of the other three algorithms (part (a)). There are only 211 audio gaps during the conference and the average audio latency is much better than with BL, VSO, or TSO. Video fidelity is also much better, with controlled video degradation during the periods with high network congestion and better video performance during periods of low congestion. There are many video gaps (*i.e.*, 3,216), but fewer than with BL, VSO, or TSO, and with much less message loss. The fidelity of the media streams is inferior to that delivered in the previous experiment using RS (Figure 5-8), but this is directly due to the MTU change on the middle segment. In particular, in the previous experiment adaptation of the audio message rate was usually sufficient to address the access constraint, but now fragmentation increases the number of packets on the middle network and forces RS to make additional adaptations to address the access constraint. In the previous experiments where the middle token ring had an MTU of 17,800 bytes, RS is able to ameliorate the access constraint by packaging audio and only making brief changes to

the video coding quality and frame rate (see Figure 5-8 (e) and (f)). In this experiment, RS not only packages audio, but must also reduce the video coding quality and frame rate to adapt to the constraint (see Figure 5-13 (e) and (f)).

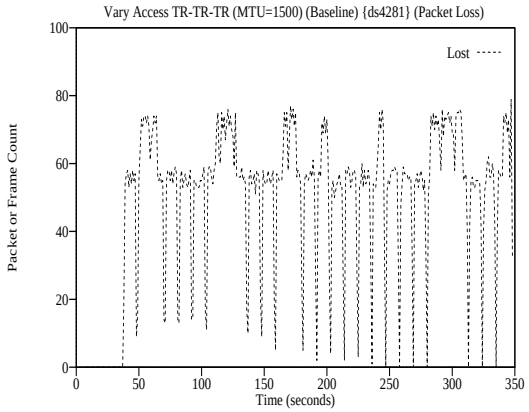
Nevertheless, even in the more hostile environment of the second experiment, RS produces a conference with good audio fidelity, controlled video fidelity, low message loss (see (b) and Table 5-5) and generally low stream latencies (parts (c) and (d)). Furthermore, RS significantly outperforms the BL, VSO, and TSO algorithms in this environment. RS achieves these results without being able to directly set the packet rate on the middle token ring (since fragmentation occurs at *R1*). RS is successful even though it can only control the message rate at the sender. Fragmentation exacerbates the access constraint, but does not prevent RS from adapting to the access constraint.

5.4.2.3. Token Ring Fragmentation Conclusions

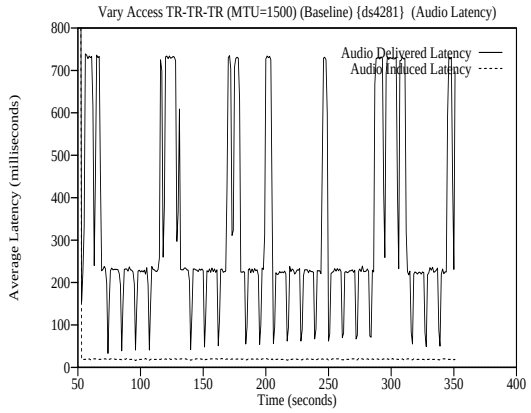
The intent of this experiment is to demonstrate the negative effect fragmentation can have in an access constrained network. The only difference between the first set of access constrained network experiments (Figures 5-5 through 5-8) and the set of experiments in this section (Figures 5-10 through 5-13) is the MTU on the middle token ring segment of Figure 5-3, yet the results of the experiments are very different. In particular, the delivered conference quality for each of the transmission control algorithms is consistently worse when the MTU is small. The message fragmentation required by the small MTU accentuates the access constraint. The BL and VSO algorithms do not perform well in either experiment. The TSO algorithm performs better than BL and VSO, but conference quality is still poor and gets much worse in the fragmented experiment. The RS algorithm has more flexibility responding to access constraints and outperforms the other three algorithms in both the unfragmented and fragmented experiments. In the unfragmented experiment, RS is generally able to deal with congestion solely through audio packaging. In the fragmented environment, RS must reduce the video message rate in addition to the audio message rate to ameliorate the effects of the access constraint, with a corresponding drop in video quality. Nevertheless, audio quality is preserved and the delivered video quality varies with the network congestion. RS is far superior at controlling message loss in both environments than the other algorithms and message loss remains low even in the more hostile fragmented environment.



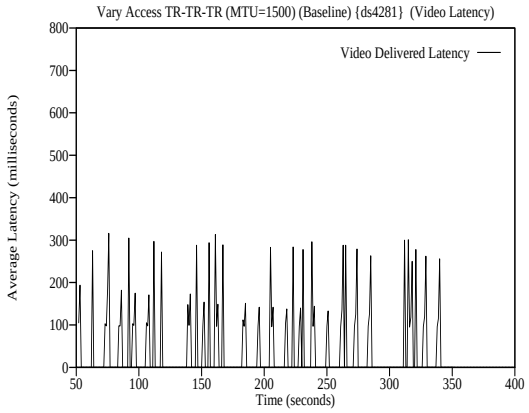
(a) Frames Per Second (FPS)



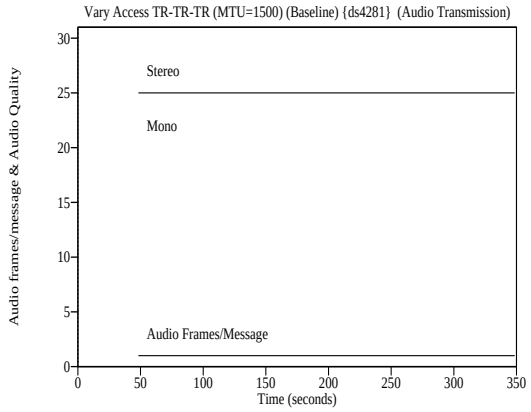
(b) Message Loss



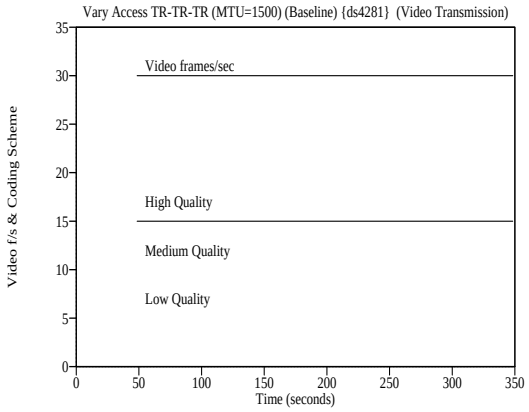
(c) Audio Latency



(d) Video Latency

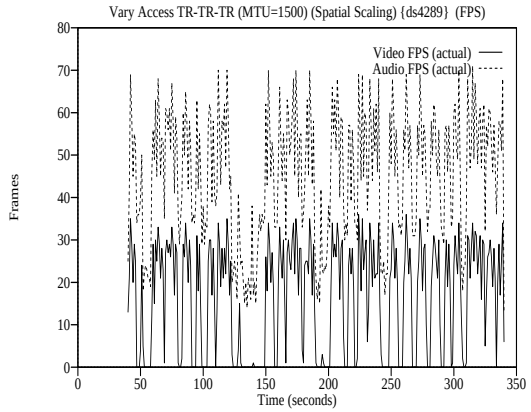


(e) Audio Transmission

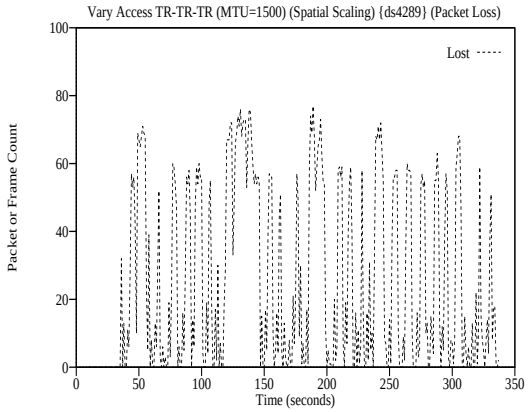


(f) Video Transmission

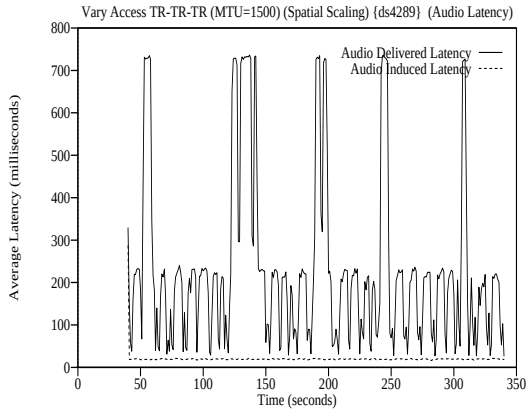
Figure 5-10: Access Constraint (TR-TR-TR; MTU=1,500) - Baseline



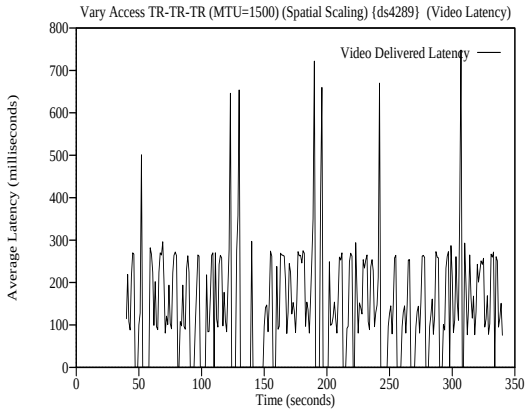
(a) Frames Per Second (FPS)



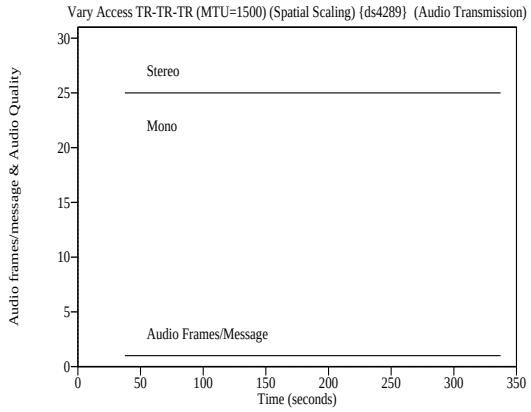
(b) Message Loss



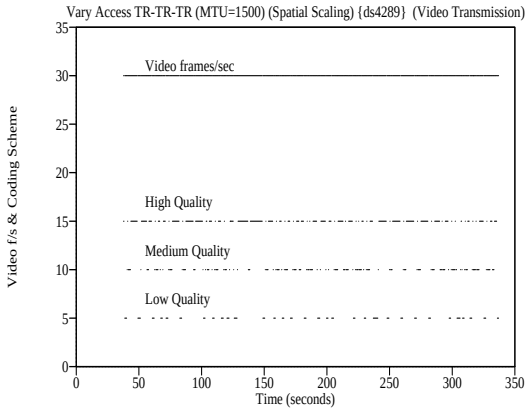
(c) Audio Latency



(d) Video Latency

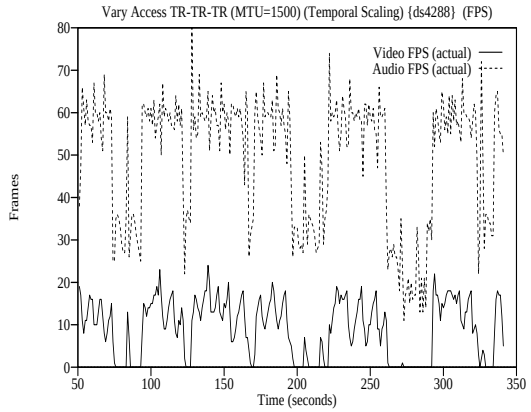


(e) Audio Transmission

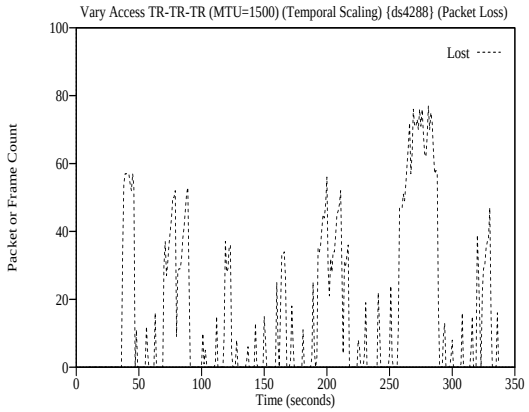


(f) Video Transmission

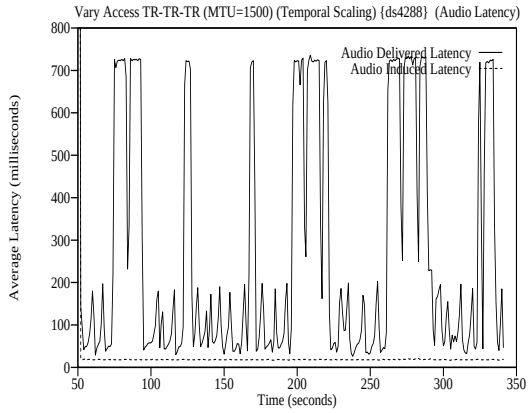
Figure 5-11: Access Constraint (TR-TR-TR; MTU=1,500) - Video Scaling Only



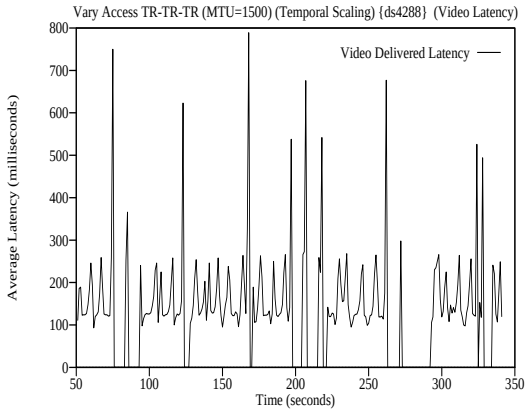
(a) Frames Per Second (FPS)



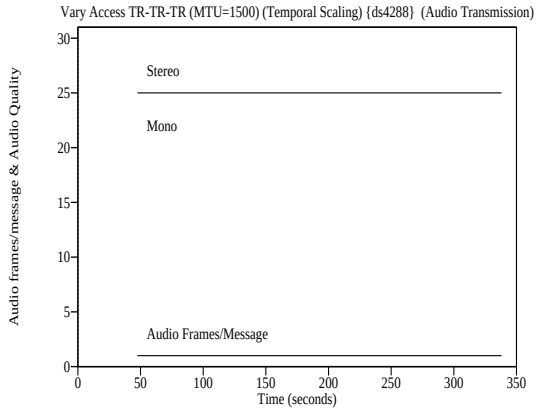
(b) Message Loss



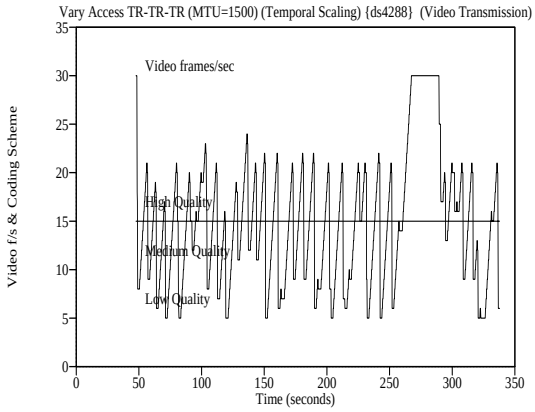
(c) Audio Latency



(d) Video Latency

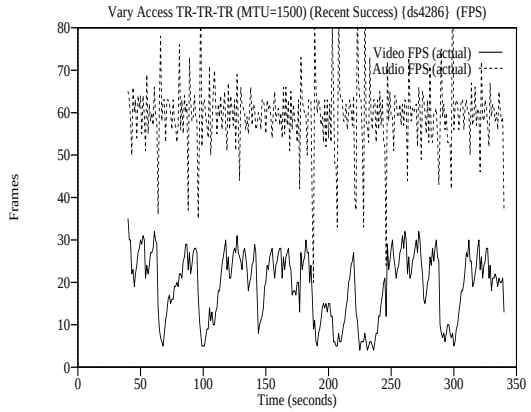


(e) Audio Transmission

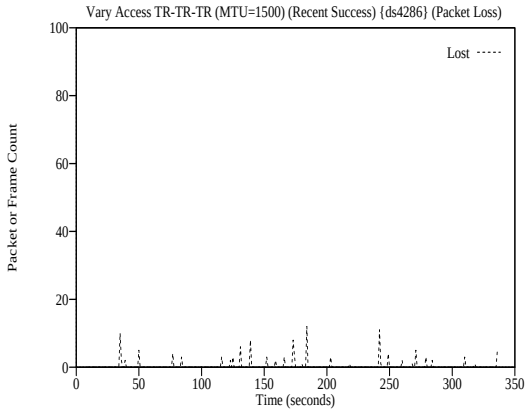


(f) Video Transmission

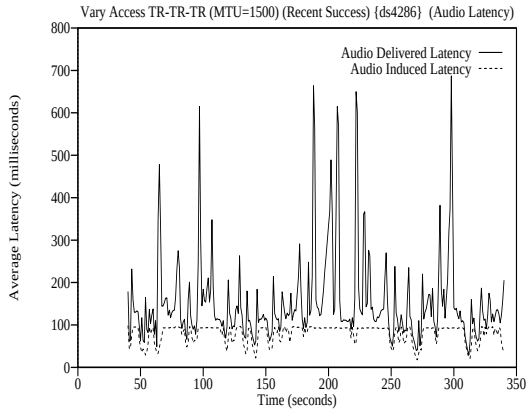
Figure 5-12: Access Constraint (TR-TR-TR; MTU=1,500) - Temporal Scaling Only



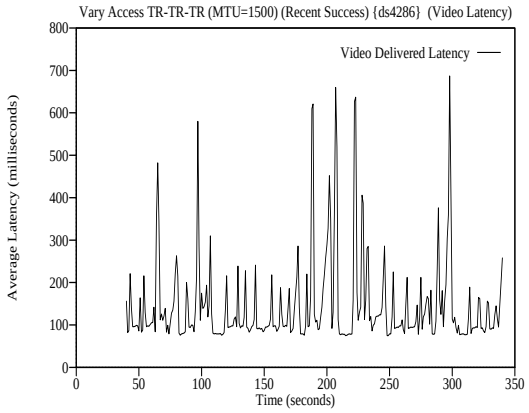
(a) Frames Per Second (FPS)



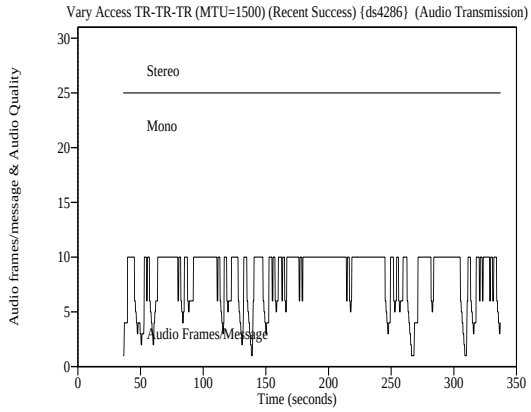
(b) Message Loss



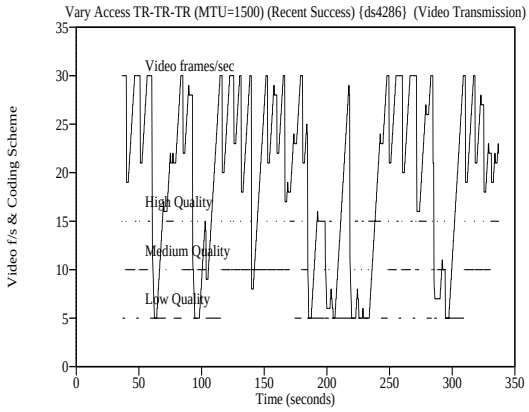
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-13: Access Constraint (TR-TR-TR; MTU=1,500) - Recent Success

Access MTU=1500	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	32.23	45.19	49.17	59.38
Standard deviation	11.56	14.92	15.10	8.72
Minimum	10	13	11	20
Maximum	67	71	80	102
Mode/Median	34/34	58/47	60/57	63/60
Gaps	8544	4419	3382	211
Audio Latency: Delivered (Induced)				
Mean	322.04 (19.25)	231.59 (19.44)	267.24 (18.36)	154.60 (79.56)
Standard deviation	219.57 (0.87)	205.47 (0.90)	279.21 (0.66)	106.43 (20.73)
Minimum	33 (16)	27 (16)	12 (17)	30 (18)
Maximum	739 (21)	736 (21)	736 (21)	687 (97)
Mode/Median	226/229 (19/19)	220/208 (19/19)	723/114 (18/18)	111/123 (93/92)
Intervals > 250 ms	85	53	91	32
Audio Messages				
Frames Sent	18601	18045	18028	18105
Msgs(frames) Lost	8531 (8531)	4393 (4393)	3355 (3355)	28 (159)
Mean Frames Lost	27.43	14.55	11.15	0.52
Max Frames Lost	49	47	47	30
Video FPS				
Mean	3.15	16.51	9.09	19.27
Standard deviation	7.73	12.88	6.80	7.90
Minimum	0	0	0	4
Maximum	30	36	24	32
Mode/Median	0/0	0/20	0/10	27/21
Gaps	8311	4044	6341	3216
Video Latency: Delivered				
Mean	41.09	143.82	132.87	145.67
Standard deviation	84.12	132.01	122.94	106.48
Minimum	0	0	0	75
Maximum	316	748	789	687
Mode/Median	0/0	0/121	0/124	79/100
Intervals > 250 ms	23	78	31	32
Video Messages				
Frames Sent	9300	9023	4646	5945
Frames Lost	8303	4030	1964	100
Mean Frames Lost	26.70	13.34	6.52	0.33
Max Frames Lost	31	31	30	9

Table 5-5: Varying Access Constraint with MTU=1500 Experiment Summary

5.4.3. Access Constraints on Ethernets

5.4.3.1. Test Environment

The experiments in this section show the results of conferences using the Baseline (BL), Video Scaling Only (VSO), and Recent Success (RS) transmission control schemes on a network that uses an Ethernet to connect two token rings (see Figure 5-14). We attached traffic generators (*P1-P7*) to the Ethernet network.

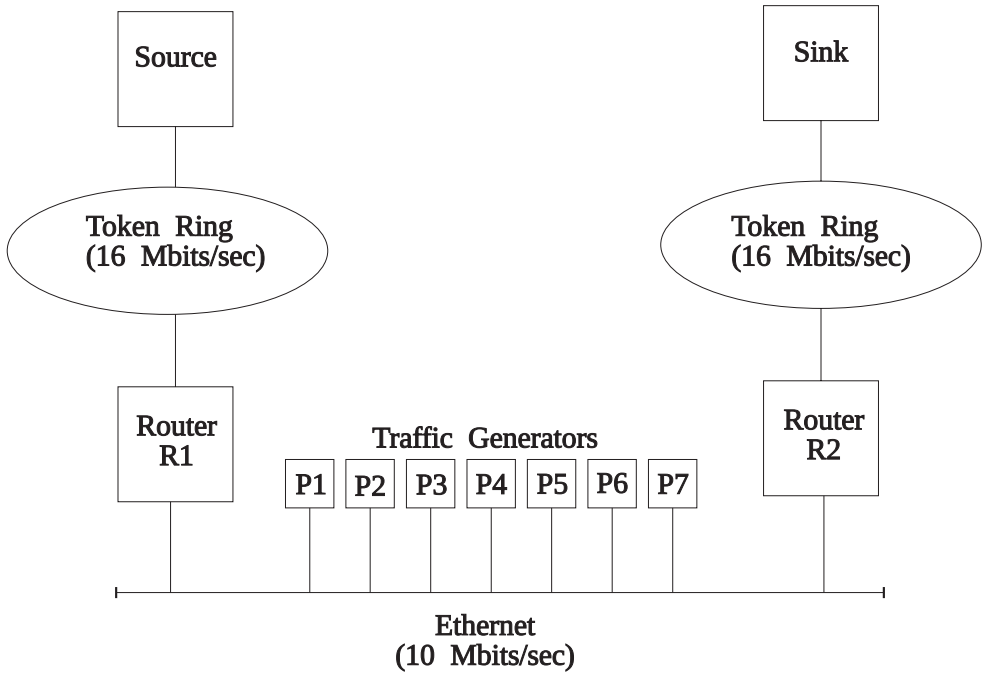


Figure 5-14: Ethernet Backbone Network

Table 5-6 shows the traffic scripts for the generators. The experiments in this section and in the later section on combination constraints use one of the four combinations of traffic generators in Table 5-7. The competition among the traffic generators for access to the Ethernet and the resultant backoffs and collisions generate an access constraint on the Ethernet. The effect of the combinations of traffic generators and scripts is to generate varying degrees of access constraint on the Ethernet. Traffic on the Ethernet varies from light to heavy, with corresponding congestion at *R1* when traffic is heavy. As with the token ring experiments, the intent of the experimental traffic pattern is not to mimic any particular traffic experienced on a production LAN, but to instead illustrate the effects of a severe access constraint on the quality of a

conference. Ethernet cannot sustain the same maximum load as token rings (the transmission rate is 10 Mbits/second for Ethernet compared with 16 Mbits/second for token ring), so although there are more traffic generators than in the token ring experiments, the actual offered loaded is less. This experiment again illustrates that only a few active machines are required to create significant congestion problems. The experiment also shows the impact of the secondary effects of video scaling on video packaging. If video scaling reduces the bit rate of the video stream enough, the packet rate resulting from the video stream will eventually be lower. The reduction in packet rate may be enough to offset the effects of moderate access constraints. In some environments, this phenomenon can make video scaling competitive with the RS algorithm. However, we show later in this chapter that under heavy loads and with more flexible conferencing systems, RS produces better results than any of the transmission schemes evaluated, including video scaling.

Script 1	Script 2	Script 3
1 size constant 1500 1 wait normal 0.01, 0.009 1 duration constant 120	1 size constant 1500 1 wait normal 0.002, 0.0015 1 duration 120	1 size normal 750, 750 1 wait normal 1.0, .75 1 duration normal 27.0, 17.0 2 size constant 1500 2 wait normal 0.004, 0.005 2 duration normal 19.0, 9.0

Table 5-6: Ethernet Traffic Generator Scripts

	Access Load 1	Access Load 2	Access Load 3	Access Load 4
Generator 1	Script 1	Script 1	Script 1	Script 1
Generator 2	Script 1	Script 1	Script 1	Script 1
Generator 3	Script 1	Script 1	Script 1	Script 1
Generator 4	Script 1	Script 2	Script 1	Script 1
Generator 5	Script 2	Script 3	Script 2	Script 1
Generator 6	Script 3	Script 3	Script 3	Script 2
Generator 7	N/A	N/A	Script 3	Script 3

Table 5-7: Traffic Generator Configurations

5.4.3.2. Experimental Results

Figures 5-15, 5-16, and 5-13 show the results from video conferences using the BL, VSO, and RS algorithms, respectively, on the network in Figure 5-14. The router *R1* is access constrained because of the competition for access to the Ethernet network segment resulting from using the traffic generator configuration Access Load 1 (Table

5-7).[4] In many ways, this experiment is similar to the previous experiments using the network in Figure 5-3 with an MTU of 1,500 bytes on the middle token ring network. The primary differences are the traffic loads and the protocol for acquiring control of the medium. The traffic loads are different because 16 Mbits/second token ring networks can support higher data rates than 10 Mbits/second Ethernet networks. Medium access is also different because token rings use a token passing protocol for controlling access, while Ethernet uses the CSMA/CD protocol. Medium access times on token passing networks are inherently more predictable than on networks using CSMA/CD, so we expect more variability of results on the network containing the Ethernet segment.

Figure 5-1 shows the operating points for the conferencing system used in this experiment. Later in this section we will use another conferencing system with a lower overall bit rate. To distinguish between the systems, we refer to the system described in Figure 5-1 as the *High Bit Rate (HBR)* system. This is a the same conferencing system used in the token ring access constrained experiments earlier in this chapter, with identical algorithms, thresholds, and potential operating points. Figure 5-9 shows the realization of the operating points on the Ethernet network, which is identical to the realization when the token ring logical MTU is 1,500 bytes. When the MTU is 1,500 bytes, video is a much larger contributor to the aggregate conference packet rate than audio (Figure 5-9). This is not the case when the MTU for the token ring 17,800 bytes (see the realization of the HBR system in Figure 5-4). Given this information, it is likely that the relative performance differences between the video scaling techniques and techniques using both scaling and packaging will be less since the secondary effects of bit rate scaling can have a significant effect upon the packet rate on the congested segment (*i.e.*, the Ethernet).[5]

---

[4]Ideally, we would measure the average time required for a station to get access to the Ethernet. We could then relate this time to the medium access time measured in the earlier token ring experiments (*e.g.*, 24 *ms* as measured by the IBM Trace and Performance tool). Unfortunately, our Ethernet traffic monitoring software does not measure average medium access time.

[5]This is also the case with token ring networks when the MTU is set to 1,500 bytes. However, as shown later in this chapter, RS increasingly outperforms video scaling as network load increases. In

To illustrate the secondary effect of bit rate scaling on the packet rate, suppose that the current video operating point is (30, 1920k). The realization of the video operating point (30, 1920k) on the Ethernet network is (30, 1920k, 180) (see Figure 5-9). In other words, 30 high quality video frames transmitted as individual messages from the conference source produces 180 packets on the Ethernet. Now suppose a video scaling algorithm such as VSO reduces the video bit rate by using the low quality video encoding rather than the high quality encoding. This corresponds to the using the operating point (30, 480k). The primary effect of this change is to cut the bit rate to 25% its original rate. The realization of (30, 480k) on the Ethernet is (30, 480k, 60), so the secondary effect of the coding scheme change is to reduce the packet rate on the Ethernet from 180 packets per second to 60 packets per second. The bit rate change has indirectly caused a packet rate reduction of 120 packets per second on the Ethernet. This change in packet rate may be enough to ameliorate the effects of the access constraint on the Ethernet.

Figure 5-15 shows that the performance of the BL algorithm is poor during congested periods on the network. There are many intervals when the delivered audio and video frame rates are low for extended periods of time (part (a)). During the congested periods, message loss (part (b)) and media latencies (parts (c) and (d)) rise sharply.

Figure 5-16 shows the performance when the conference uses the VSO algorithm for transmission control. VSO dynamically alters the coding scheme of video based on the network conditions (part (f)), which results in significantly better conference performance when compared with the results obtained with BL. Audio and video frame rates (part (a)) are better than with BL. There are fewer audio and video gaps (711 audio gaps versus 2,988 with BL and 403 video gaps compared with 1,767 under BL; Table 5-8). The delivered audio and video frame rates suffer to approximately the same degree during periods of congestion. Media latencies are lower than those delivered with BL. There is less message loss (part (b)) than with the BL algorithm, although significant message loss still occurs.

---

the case of the earlier token ring experiments with MTU=1,500 bytes, the access constraint is severe enough that RS significantly outperforms video scaling.

Figure 5-17 shows the conference performance under the RS algorithm.[6] Audio frame delivery is better than with either BL or VSO. With RS, there are only 267 audio gaps and the average delivered audio frame rate is over 59 frames per second. Video fidelity is not as good with RS as with VSO. The conference has a lower delivered video frame rate under RS (20.59) than with either the BL (24.30) or the VSO (28.40) algorithms. This is a direct result of RS intentionally lowering the video message rate during congested periods. In effect, video fidelity is sacrificed for improved audio performance. Since video is sacrificed at the source (*i.e.*, with lower generated video frame rates), RS experiences lower message loss (part (b)) than either BL or VSO and has lower video latency. Audio latency is lower with VSO than with RS. RS often packages multiple audio frames into one message (part (e)). The induced latency associated with audio packaging increases the delivered audio latency. However, RS still has audio latency lower than the latency guideline of 250 milliseconds (part (c)).

5.4.3.3. Conclusions from the HBR Experiments on Ethernet

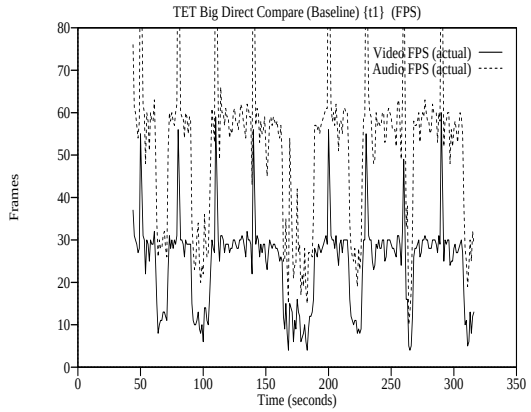
It is more difficult to identify a clear winner with this experiment than with earlier experiments. In these experiments, the relatively moderate access constraint on the Ethernet network and the fragmentation of the video frames at *R1* makes VSO much more competitive with RS than in the earlier experiments in this chapter. The reduction in bit rate resulting from video scaling causes a secondary reduction in packet rates on the congested hop that is sufficient to ameliorate effects of the access constraint. Furthermore, fragmentation of the video frames limits the ability of RS to reduce the video packet rate on the access constrained Ethernet without resorting to spatial and temporal scaling of the video stream. As a result, the VSO algorithm delivers lower audio latency and better video frame rates than RS. On the other hand, RS delivers better audio frame rates and has less message loss. From the perspective

---

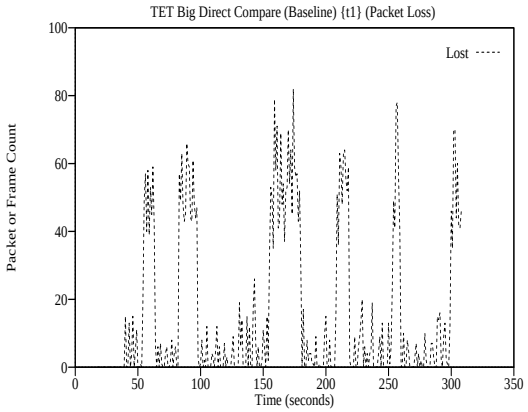
[6]We did not use the TSO algorithm in this set of experiments. We developed the TSO algorithm after these experiments were run. Ideally, we would have repeated this experiment using TSO, but we did not. Based on the previous token ring experiments with MTU=1,500, we expect TSO would have provided slightly better audio performance than VSO, but worse video performance. We expect TSO would have provided inferior performance to RS for both audio and video. Although we do not have controlled network experiments for this topology for TSO, we do have experiments using TSO in a similar Ethernet environment for the production network experiments in Chapter 6.

of conference quality, the results for VSO and RS are essentially a draw, with both better than the BL algorithm. From a network perspective, RS is superior to both VSO and BL.

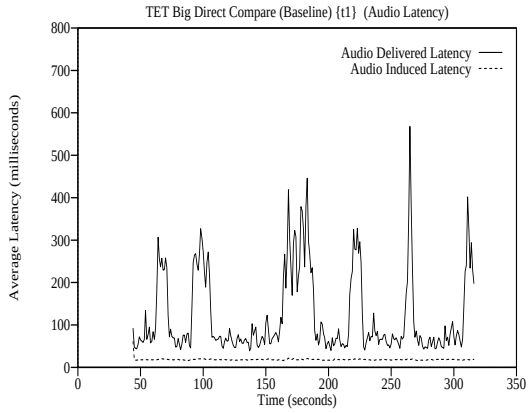
This experiment demonstrates that video scaling can deliver comparable performance to RS provided the network is only moderately congested and the network MTU is small in relation to the media frames. The bit rate reductions resulting from video scaling reduce the packet rate to the point that the access constraint is relieved. The motivation for video scaling is to reduce the bit rate, which does not address access constraints, but the eventual effect is to reduce the packet rate, which does address access constraints. In essence, video scaling becomes a two-dimensional algorithm like RS. As such, video scaling can have some degree of success ameliorating the effects of both capacity and access constraints. However, later experiments demonstrate that when access constraints are severe or the network MTU is large relative to the media frames, video scaling is inferior to RS. Even when video scaling is competitive with RS, RS still provides better audio throughput and lower message loss rates than with VSO. VSO only outperforms RS for video throughput and delivered audio latency, and then only in certain environments. Note that the RS algorithm used here is operating under a latency budget of 250 milliseconds for the audio stream. This budget is an input to the RS algorithm and directly affects the amount of induced latency the RS algorithm will accept. It is likely that RS will deliver lower audio latencies if we lower the latency budget. VSO does not consider the latency budget, so changing the budget does not change the performance of the VSO algorithm.



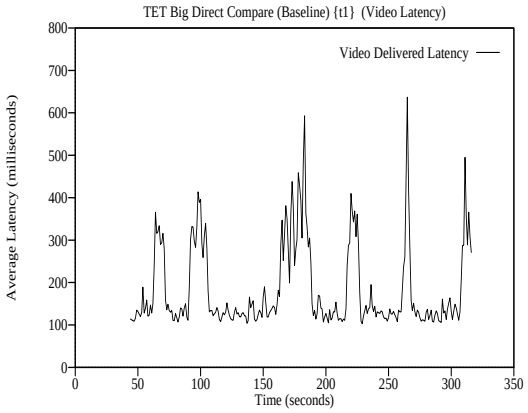
(a) Frames Per Second (FPS)



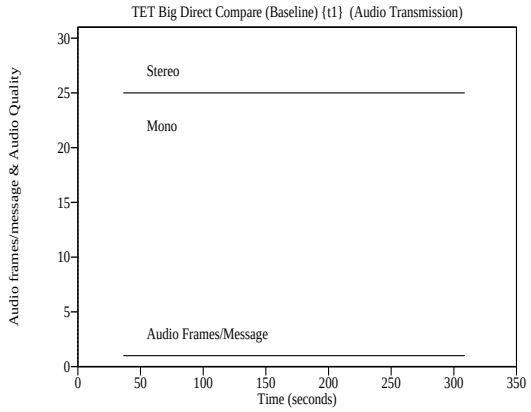
(b) Message Loss



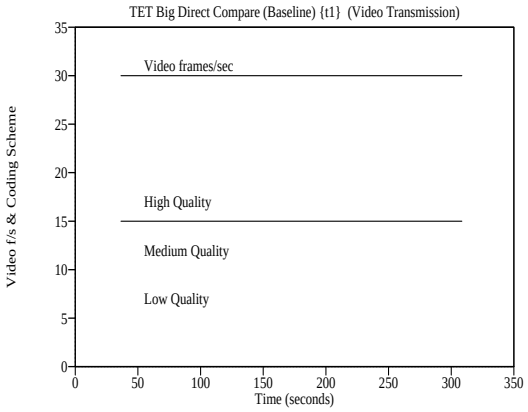
(c) Audio Latency



(d) Video Latency

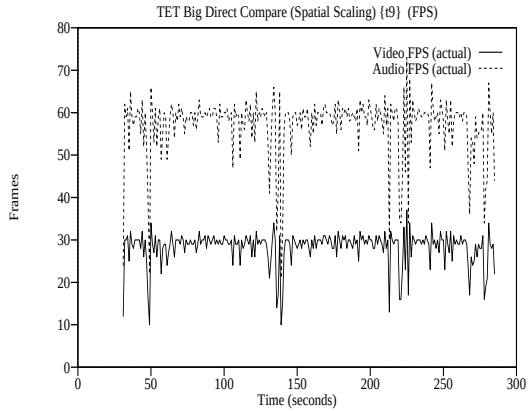


(e) Audio Transmission

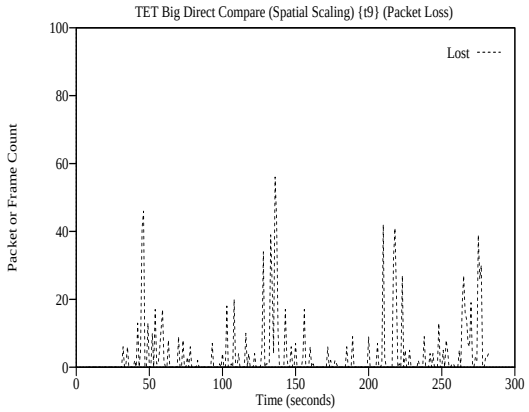


(f) Video Transmission

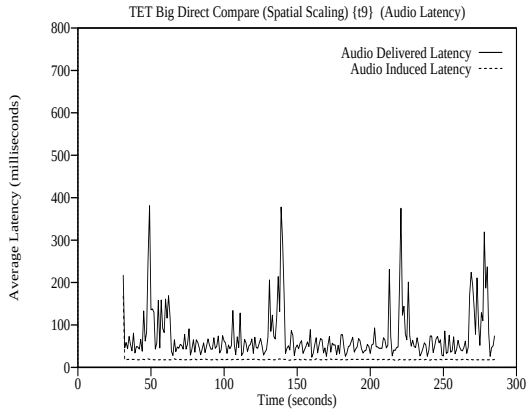
Figure 5-15: Access Constraint (TR-EN-TR; High Bit Rate System) Experiment - Baseline



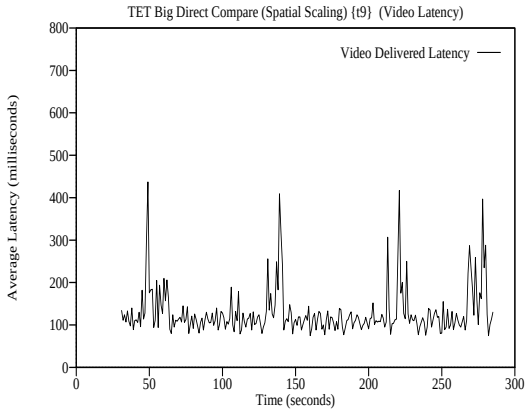
(a) Frames Per Second (FPS)



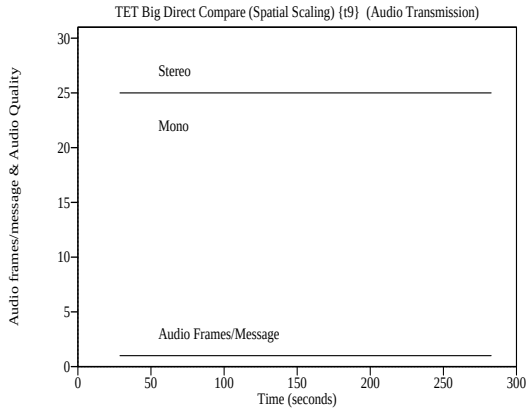
(b) Message Loss



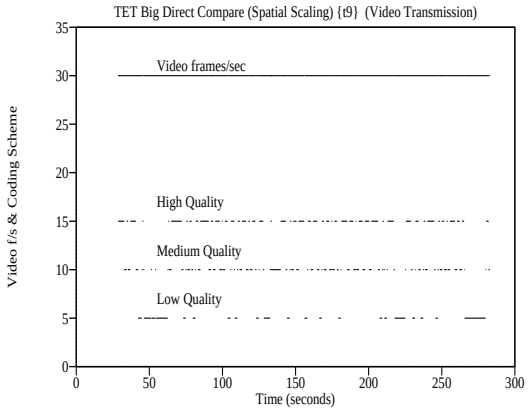
(c) Audio Latency



(d) Video Latency

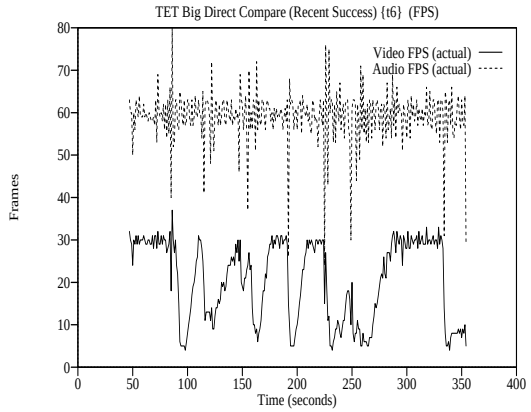


(e) Audio Transmission

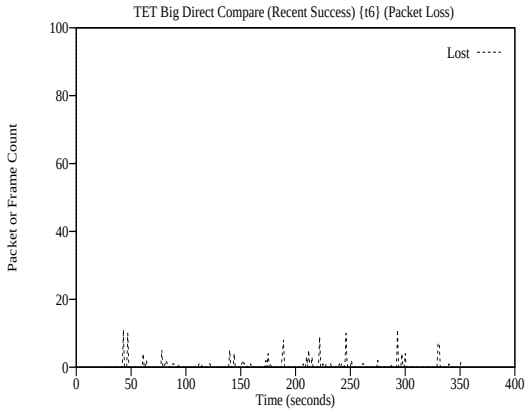


(f) Video Transmission

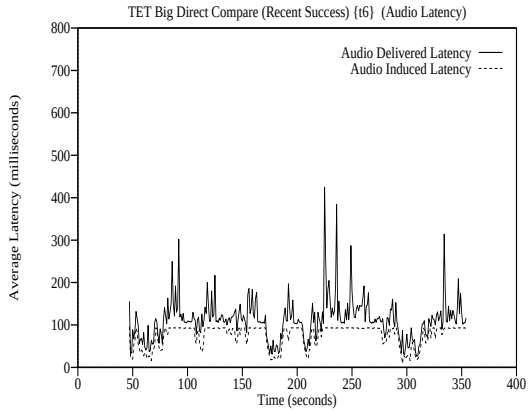
Figure 5-16: Access Constraint (TR-EN-TR; High Bit Rate System) Experiment - Video Scaling Only



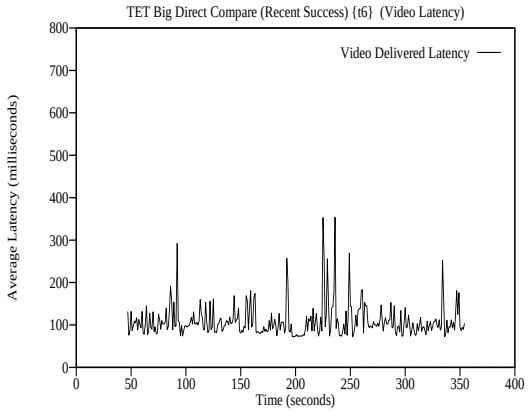
(a) Frames Per Second (FPS)



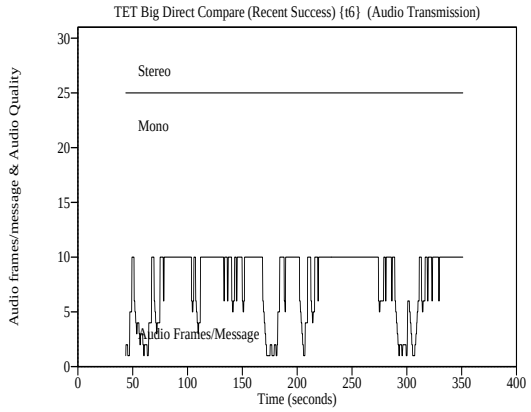
(b) Message Loss



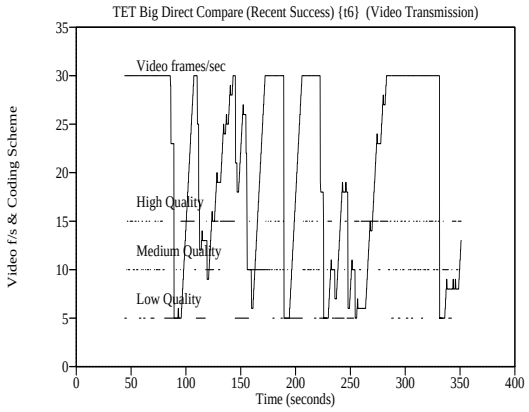
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-17: Access Constraint (TR-EN-TR; High Bit Rate System) Experiment - Recent Success

TR-EN-TR HBR	Baseline	VSO	Recent Success
Audio FPS			
Mean	50.64	57.17	59.10
Standard deviation	18.12	7.07	6.33
Minimum	10	21	24
Maximum	120	73	81
Mode/Median	60/57	60/59	63/60
Gaps	2988	711	267
Audio Latency: Delivered (Induced)			
Mean	123.09 (18.31)	72.44 (18.58)	112.32 (75.27)
Standard deviation	98.31 (0.95)	58.49 (0.69)	48.98 (25.43)
Minimum	39 (16)	24 (16)	26 (9)
Maximum	568 (22)	381 (20)	425 (94)
Mode/Median	73/73 (18/18)	43/53 (19/19)	107/109 (93/92)
Intervals > 250 ms	38	7	7
Audio Messages			
Frames Sent	16395	15263	18461
Msgs(frames) Lost	2978 (2978)	690 (690)	198 (228)
Mean Frames Lost	10.87	2.71	0.74
Max Frames Lost	54	36	30
Video FPS			
Mean	24.30	28.40	20.59
Standard deviation	10.02	3.88	9.50
Minimum	4	10	4
Maximum	60	37	37
Mode/Median	30/28	30/29	30/24
Gaps	1767	403	2875
Video Latency: Delivered			
Mean	186.22	128.97	109.41
Standard deviation	101.09	56.28	39.21
Minimum	103	75	72
Maximum	637	437	354
Mode/Median	131/135	95/113	74/99
Intervals > 250 ms	66	13	8
Video Messages			
Frames Sent	8198	7632	6478
Frames Lost	1760	392	112
Mean Frames Lost	6.42	1.54	0.36
Max Frames Lost	29	20	8

Table 5-8: Varying Access Constraint with HBR System Experiment Summary

5.4.3.4. Experiments with A Low Bit Rate System

The operating points for the system used in the previous experiment have relatively high bit rates (see Figure 5-1). The MTU of the constrained network segment (*i.e.*, the Ethernet segment) is small compared with the frame size of the media streams, particularly for video frames. The following experiments compare the relative performance of the BL, VSO, and RS algorithms with a video conference system with lower bit rate requirements. Figure 5-18 shows the operating points for this system. We refer to this system as the *Low Bit Rate* (LBR) system.[7] Figure 5-19 shows the realization of the LBR operating points on Ethernet. We refer to the system used in the previous experiments (Figure 5-1) as the *High Bit Rate* (*HBR*) system.

The LBR system has two audio coding schemes: stereo audio and monaural audio. The HBR system had only one audio coding scheme (*i.e.*, stereo audio). The HBR system has three video coding schemes. The LBR system has only two video coding schemes and the bit rates resulting from these two schemes are much lower than with the HBR system (compare Figures 5-1 and 5-18). The lower video bit rates associated with LBR produce lower packet rates on the Ethernet segment than the larger video frames in the HBR system. With LBR, the video stream does not dominate the packet rate on the Ethernet to the degree seen with the HBR system (compare the packet rates on Figure 5-9 and Figure 5-19).

---

[7]The DVI system used in these experiments cannot produce the LBR video operating points. We simulated these operating points by truncating the DVI video frames to match the bit rates associated with the LBR operating points. These truncated frames were acquired, transmitted, received, and queued for display just like regular DVI video frames. The frames were not actually displayed since from a DVI perspective they were incomplete (and thus unplayable) frames. All reported frame rates, gaps, and latencies were collected as if the truncated frames were valid, playable frames.

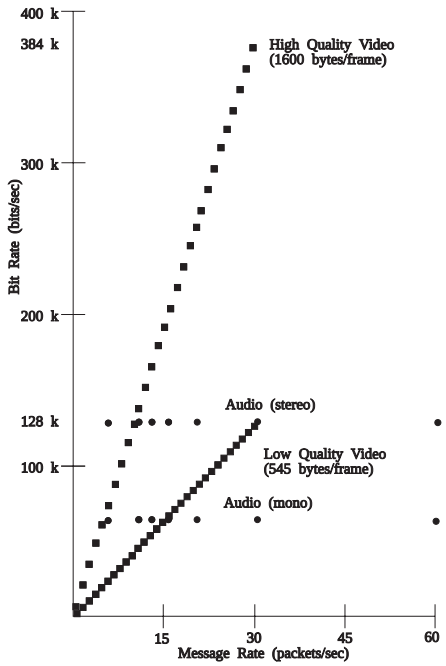


Figure 5-18: Low Bit Rate System Operating Points

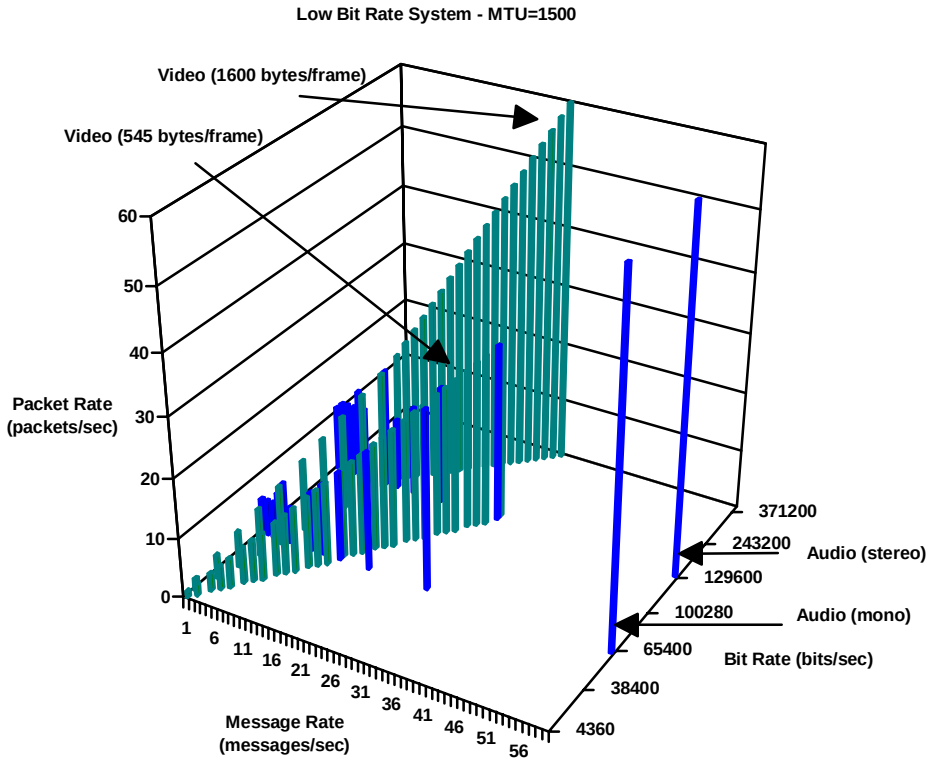


Figure 5-19: Realization of Low Bit Rate System Operating Points on Ethernet

With the LBR system, the BL algorithm uses the video operating point (30, 384k) and the audio operating point (60, 128k). The VSO algorithm uses an audio operating point (60, 128k) and can choose between either (30, 384k) or (30, 130k) for the video operating point. The RS algorithm may choose any operating point in Figure 5-18. With the HBR system, RS could scale or package video, but could only manipulate the packaging of audio. With the LBR system, RS can not only package audio, but also scale the audio stream by selecting either stereo or monaural data.

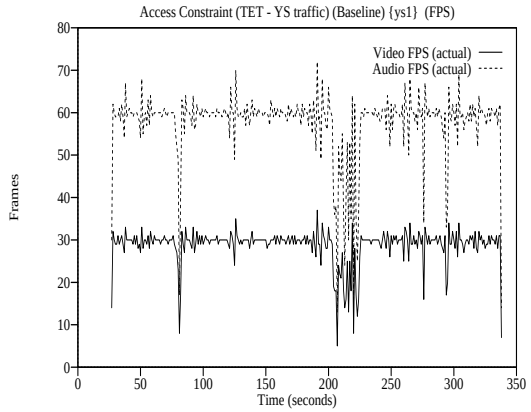
5.4.3.4.1. Moderate Congestion and the LBR System

Figures 5-20, 5-21, and 5-22 show the results for a set of video conferences using the BL, VSO, and RS algorithms, respectively. Table 5-9 summarizes these results. These experiments used the network in Figure 5-14. The traffic load (Access Load 2 in Table 5-7) in this experiment generates occasional periods of congestion. These experiments use the LBR operating points. Figure 5-20 shows that the non-adaptive transmission scheme BL does not perform well during the congested periods. The delivered audio and video frame rates drop during congested periods (part (a)). There is a dramatic increase in stream latency when the network becomes congested (parts (c) and (d)). There are also many messages lost when the network is congested (part (b)). The relatively high latency, low fidelity periods may last for several seconds (*e.g.*, for about 30 seconds starting around 190 seconds into the conference).

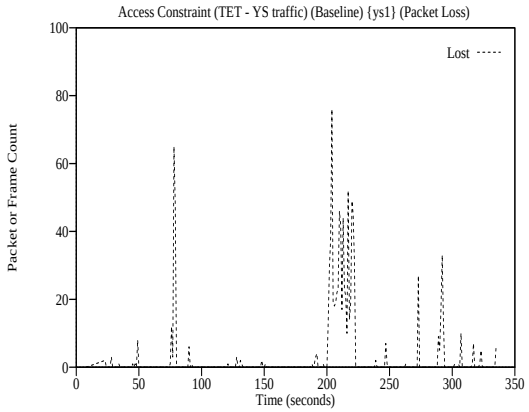
The VSO algorithm reduces the duration of the periods with poor frame delivery rates (Figure 5-21 (a)), but still experiences short periods with high latency (parts (c) and (d)). Lowering the quality of the video encoding (part (f)) reduces the aggregate packet rate and has some effect on the access constraint generated by the Ethernet load, but the reduction does not completely counteract the effects of the network congestion.

Figure 5-22 shows the results obtained with the RS algorithm. With RS, there are fewer audio gaps (146 with RS compared with 666 with BL and 520 with VSO; Table 5-9). The average audio latency is comparable to that with BL and VSO, even with the induced latency associated with audio packaging (part (c)), but RS does not experience the periods of high latencies seen with BL and VSO (parts (c) and (d)). With RS, video frame rates are generally high except at the initial onset of congestion when RS intentionally reduces the video frame rate in response to the congestion. The periods with reduced video frame rates are very short. As soon as the network

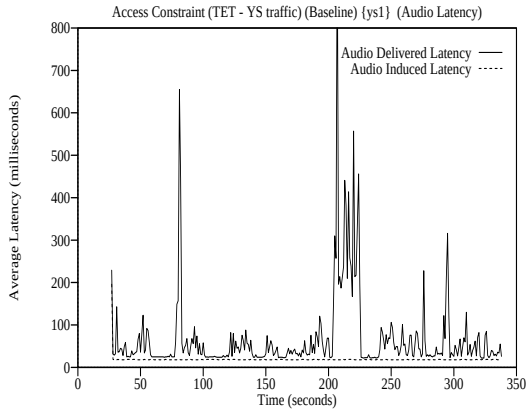
congestion is addressed, RS returns to the full video frame rate (part (a)). The average video frame rate is comparable for all three algorithms and video latency is lower with RS than with the other algorithms. There is low message loss with RS (part (b)). Parts (e) and (f) show that generally video frame rates are only reduced when adapting to congestion. During these periods, RS also packs more audio frames per message. The combination of audio packaging and the low quality video encoding are sufficient to ameliorate the effects of the access constraint on the Ethernet segment.



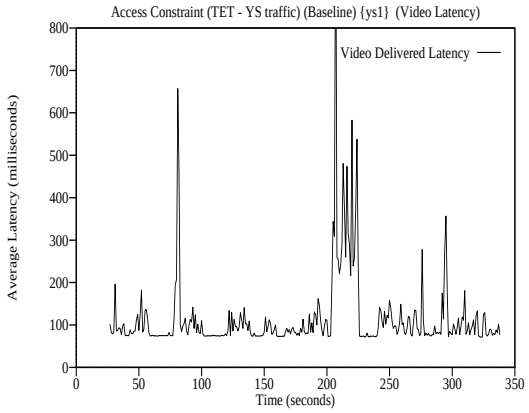
(a) Frames Per Second (FPS)



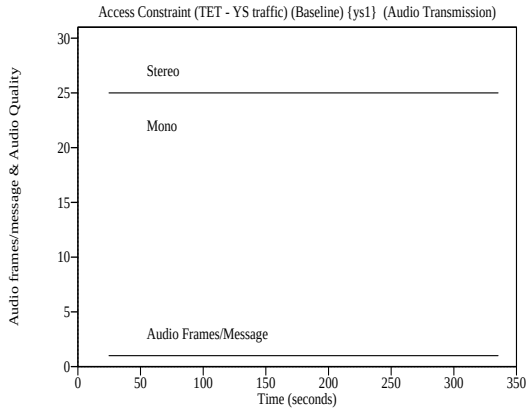
(b) Message Loss



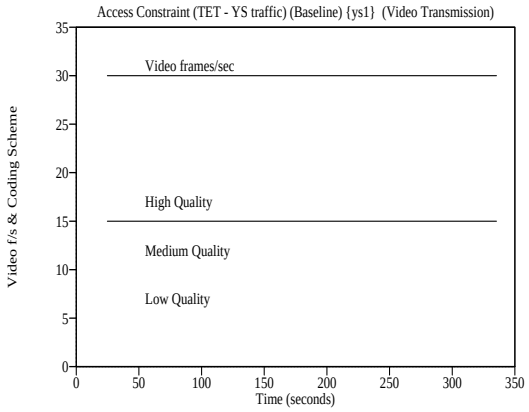
(c) Audio Latency



(d) Video Latency

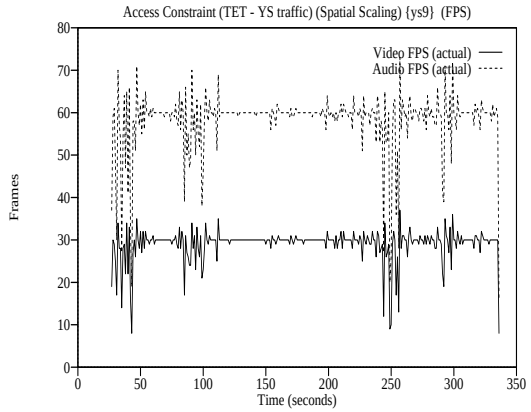


(e) Audio Transmission

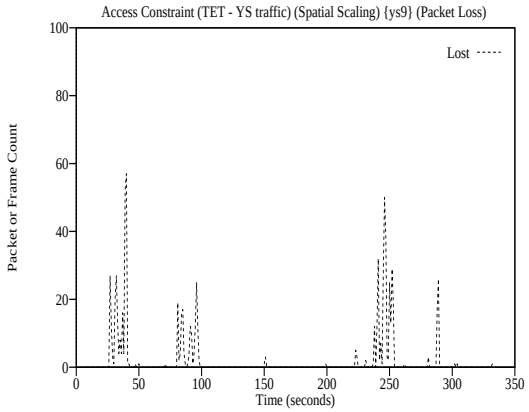


(f) Video Transmission

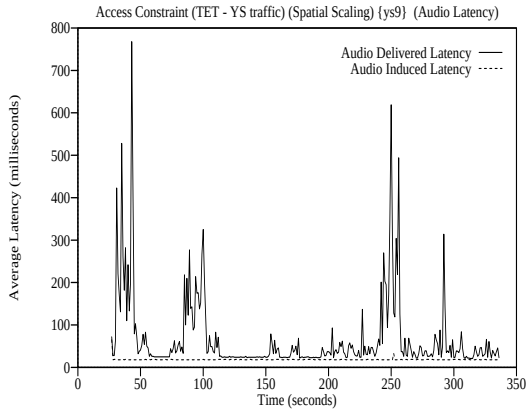
Figure 5-20: Access Constraint (TR-EN-TR; Low Bit Rate System) Experiment - Baseline



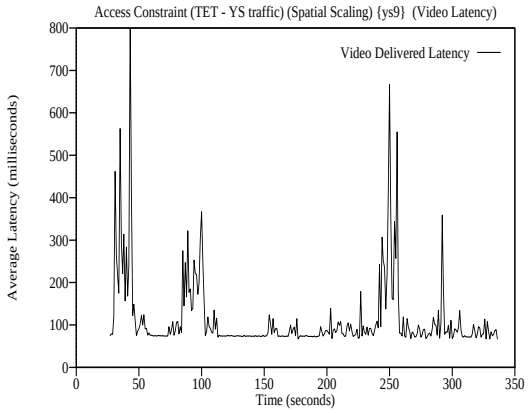
(a) Frames Per Second (FPS)



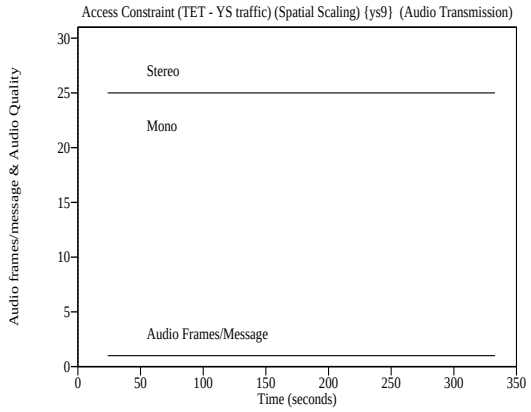
(b) Message Loss



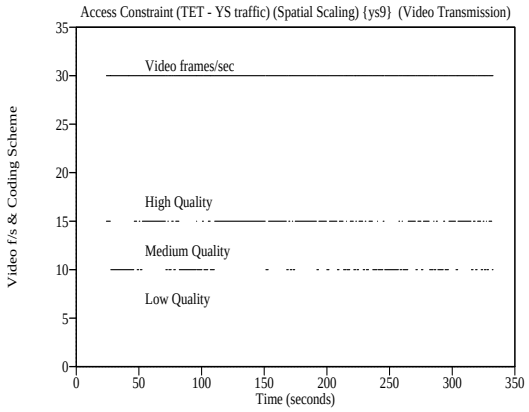
(c) Audio Latency



(d) Video Latency

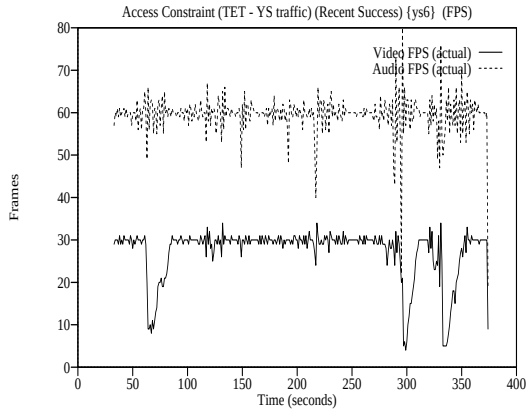


(e) Audio Transmission

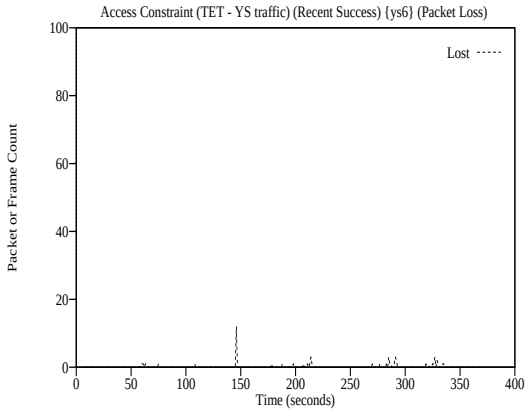


(f) Video Transmission

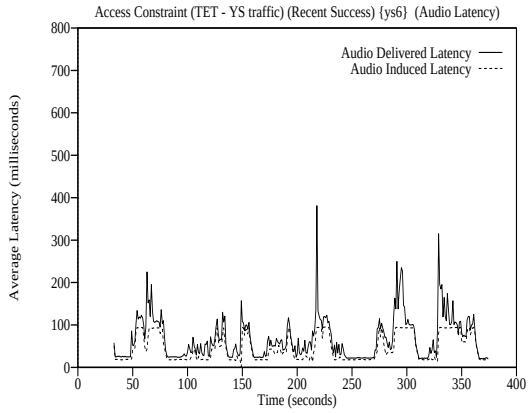
Figure 5-21: Access Constraint (TR-EN-TR; Low Bit Rate System) Experiment - Video Scaling Only



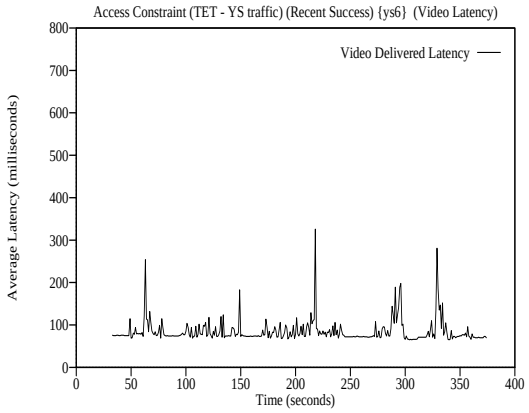
(a) Frames Per Second (FPS)



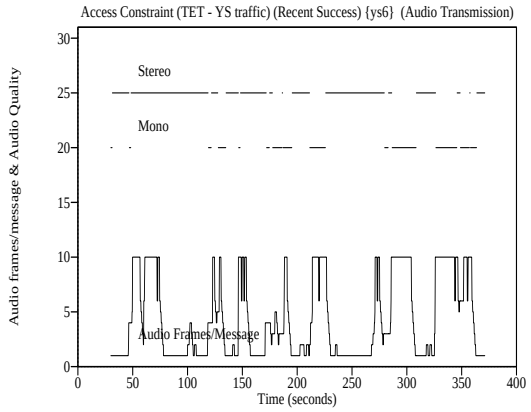
(b) Message Loss



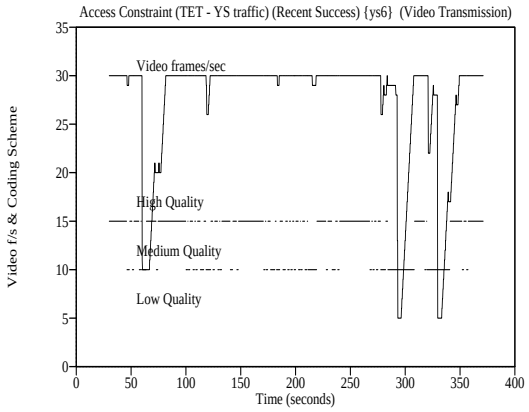
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-22: Access Constraint (TR-EN-TR; Low Bit Rate System) Experiment - Recent Success

TR-EN-TR LBR #1	Baseline	VSO	Recent Success
Audio FPS			
Mean	57.75	58.21	59.50
Standard deviation	7.99	7.32	4.79
Minimum	13	16	19
Maximum	72	73	80
Mode/Median	60/60	60/60	60/60
Gaps	666	520	146
Audio Latency: Delivered (Induced)			
Mean	67.67 (18.32)	67.84 (18.37)	67.62 (45.60)
Standard deviation	99.16 (0.47)	92.23 (0.96)	50.36 (31.22)
Minimum	22 (18)	21 (18)	20 (13)
Maximum	1033 (19)	768 (33)	381 (95)
Mode/Median	25/35 (18/18)	24/35 (18/18)	22/52 (18/31)
Intervals > 250 ms	14	17	2
Audio Messages			
Frames Sent	18663	18558	20502
Msgs(frames) Lost	640 (640)	488 (488)	23 (106)
Mean Frames Lost	2.05	1.57	0.31
Max Frames Lost	50	37	14
Video FPS			
Mean	28.79	29.08	27.29
Standard deviation	4.19	3.81	6.13
Minimum	5	8	4
Maximum	37	37	34
Mode/Median	30/30	30/30	30/30
Gaps	364	267	906
Video Latency: Delivered			
Mean	116.82	114.82	84.76
Standard deviation	99.66	91.11	27.79
Minimum	71	67	65
Maximum	1163	833	326
Mode/Median	75/86	74/81	74/75
Intervals > 250 ms	22	23	3
Video Messages			
Frames Sent	9331	9278	9391
Frames Lost	347	250	31
Mean Frames Lost	1.11	0.81	0.09
Max Frames Lost	26	20	4

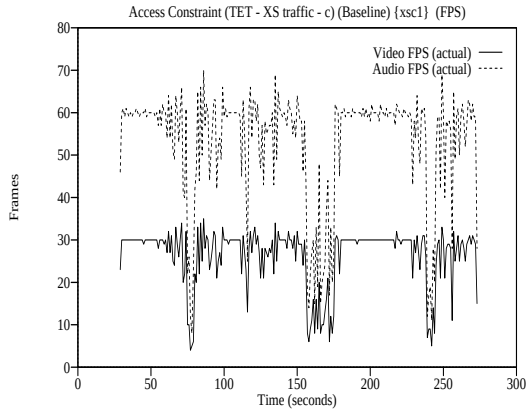
Table 5-9: Varying Access Constraint with LBR System Experiment Summary

5.4.3.4.2. High Congestion and the LBR System

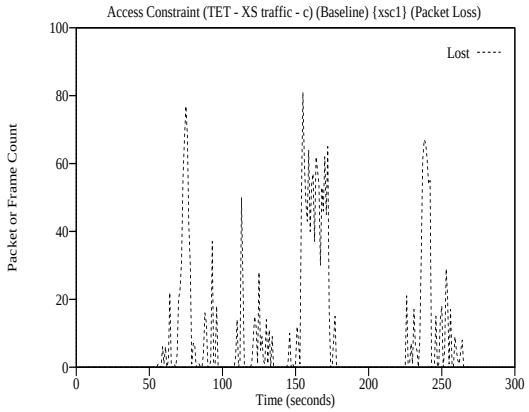
Figures 5-23, 5-24, and 5-25 show the results for conferences using BL, VSO, and RS, over the network in Figure 5-14. Table 5-10 summarizes these results. The network is more congested in this experiment than in the previous experiment (this experiment uses Access Load 3 in Table 5-7). This experiment again uses the LBR video conferencing system and illustrates how the performance differences between RS and the BL and VSO algorithms increase when the access constraints increase. Figure 5-23 shows the results obtained with the BL algorithm. Part (a) shows that there are long periods with very low audio and video frame delivery rates. Audio is often unintelligible for many seconds (*e.g.*, during the period approximately 155-175 into the conference). There are 1,644 audio gaps during the conference (Table 5-10). The frames that are delivered during the congested periods have very high latencies (parts (c) and (d)). In over a tenth of the one second reporting intervals the average latency is greater than 250 *ms*. Many messages are lost when the network is congested (part (b)).

Figure 5-24 shows that the VSO algorithm has essentially the same performance as BL under heavy congestion. There are still periods of poor audio frame delivery (part (a)), high stream latency (parts (c) and (d)), and high message loss (part (b)). The VSO conference has 2,030 audio gaps. The average delivered audio and video frame rates are virtually identical to those delivered using BL (Table 5-10). There are actually more intervals where the stream latencies exceed the 250 *ms* guideline than with BL (68 and 81 intervals for audio and video, respectively; Table 5-10). The VSO algorithm often reaches its limit on adaptation (part (f)) with video transmitted at the lowest available quality. This adaptation halves the number of video messages transmitted, but does not ameliorate the effect of the access constraint on the Ethernet.

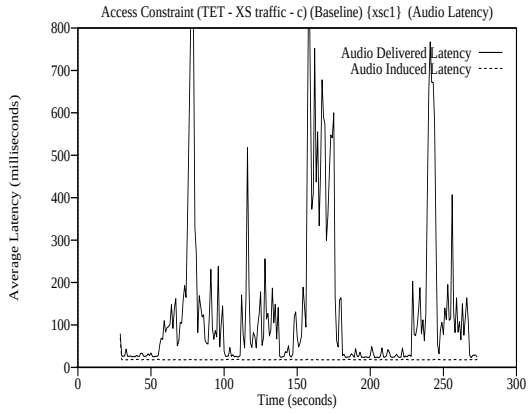
In contrast, RS scales the bit rate for both audio and video. RS also manipulates the packaging of audio frames to address the heavy access constraint. Figure 5-25 shows that although the delivered video frame rate varies according to network congestion, audio quality is preserved (part (a)). The conference using RS experiences only 147 audio gaps (Table 5-10). There are 1,831 video gaps, but the gaps result primarily from selective reductions in the video frame rate at the source. Only a relatively few messages are lost during transmission. Neither media stream ever has latency greater than 250 milliseconds (parts (c) and (d)). Overall conference quality is much higher than with either the BL or VSO algorithms.



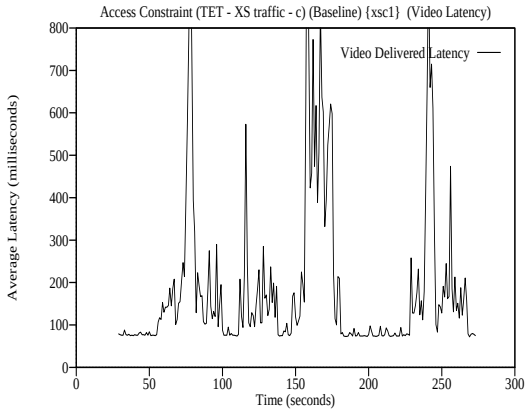
(a) Frames Per Second (FPS)



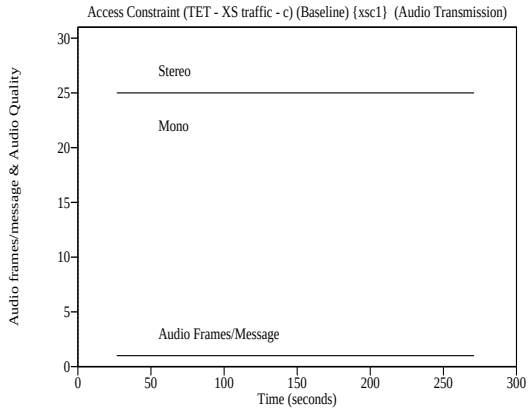
(b) Message Loss



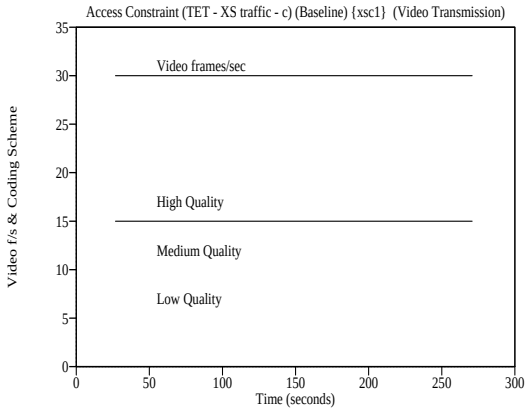
(c) Audio Latency



(d) Video Latency

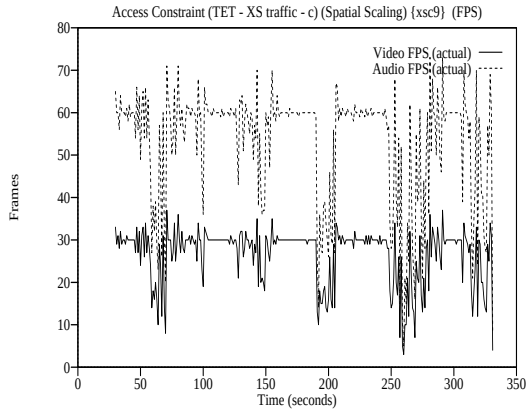


(e) Audio Transmission

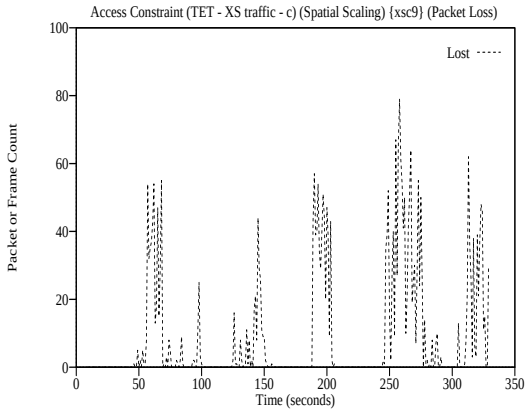


(f) Video Transmission

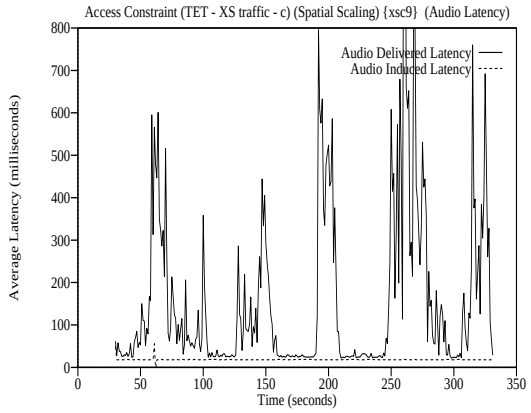
Figure 5-23: Access Constraint (TR-EN-TR; Low Bit Rate System; High Traffic) - Baseline



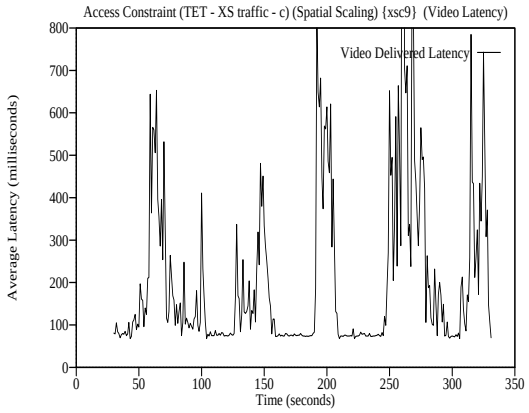
(a) Frames Per Second (FPS)



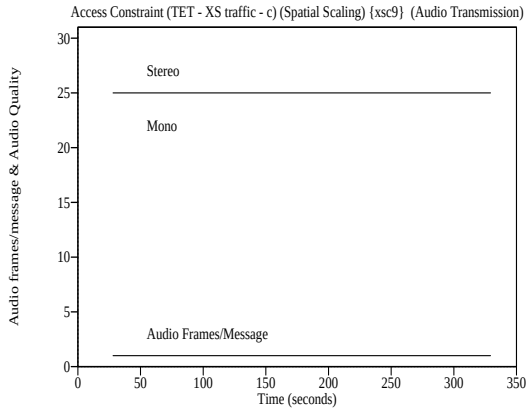
(b) Message Loss



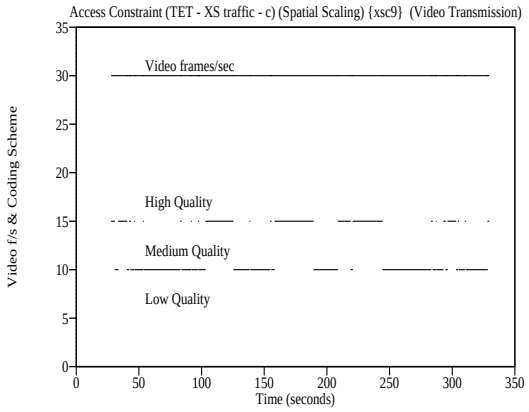
(c) Audio Latency



(d) Video Latency

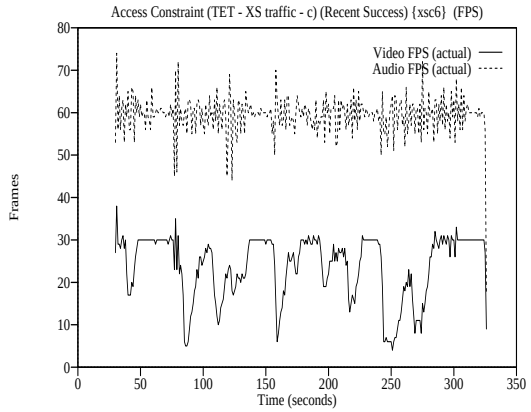


(e) Audio Transmission

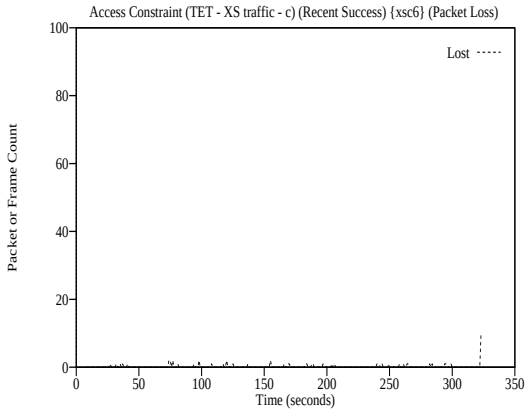


(f) Video Transmission

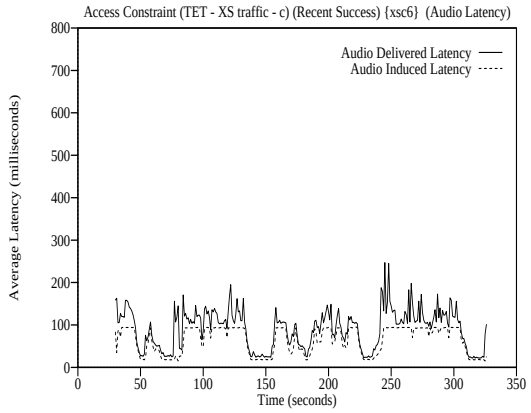
Figure 5-24: Access Constraint (TR-EN-TR; Low Bit Rate System; High Traffic) - Video Scaling Only



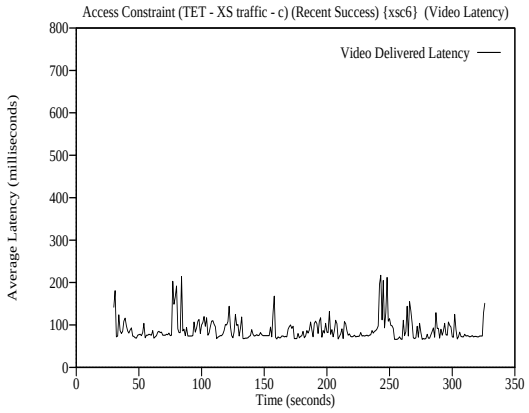
(a) Frames Per Second (FPS)



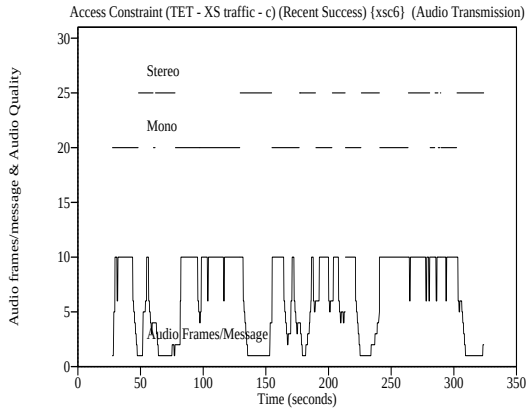
(b) Message Loss



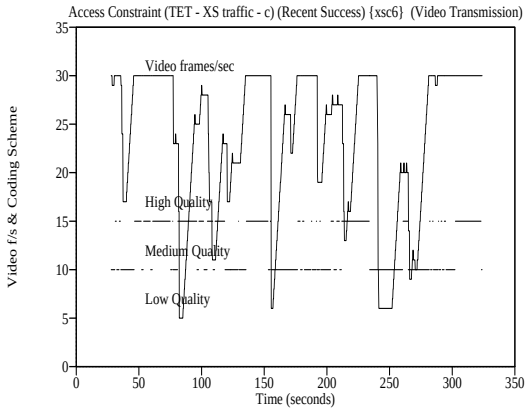
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-25: Access Constraint (TR-EN-TR; Low Bit Rate System; High Traffic) - Recent Success

TR-EN-TR LBR #2	Baseline	VSO	Recent Success
Audio FPS			
Mean	53.17	53.13	59.47
Standard deviation	13.43	13.29	4.69
Minimum	8	5	17
Maximum	70	73	74
Mode/Median	60/59	60/59	60/60
Gaps	1644	2030	147
Audio Latency: Delivered (Induced)			
Mean	139.82 (18.23)	162.12 (18.27)	91.09 (64.44)
Standard deviation	191.34 (0.42)	210.55 (2.69)	45.55 (31.57)
Minimum	23 (18)	22 (18)	22 (15)
Maximum	1108 (19)	1634 (19)	247 (95)
Mode/Median	26/65 (18/18)	25/64 (18/18)	104/104 (94/80)
Intervals > 250 ms	37	68	0
Audio Messages			
Frames Sent	14671	18119	17793
Msgs(frames) Lost	1617 (1617)	2015 (2015)	26 (119)
Mean Frames Lost	6.60	6.65	0.40
Max Frames Lost	53	52	10
Video FPS			
Mean	26.36	26.68	23.78
Standard deviation	7.01	6.57	7.73
Minimum	4	3	4
Maximum	35	37	38
Mode/Median	30/30	30/30	30/27
Gaps	879	981	1831
Video Latency: Delivered			
Mean	188.79	208.00	88.58
Standard deviation	193.16	205.15	26.47
Minimum	72	68	66
Maximum	1162	1488	217
Mode/Median	74/113	74/107	75/78
Intervals > 250 ms	40	81	0
Video Messages			
Frames Sent	7335	9059	7107
Frames Lost	864	971	31
Mean Frames Lost	3.53	3.20	0.10
Max Frames Lost	28	27	5

Table 5-10: Varying Access Constraint with LBR and Heavy Traffic Summary

5.4.3.4.3. Ethernet Access Constraint Conclusions

The experiments using the network in Figure 5-14 illustrate the potentially devastating effects access constraints may have on a video conference in an Ethernet environment. Non-adaptive algorithms such as BL are not equipped to deal with the effects of congestion. Algorithms such as VSO that only scale the video bit rate may be able to address some access constraints by reducing the video bit rate to the point that it forces a reduction in the realized packet rate. However, as congestion increases this secondary effect may be inadequate to offset the effects of access constraints. The RS algorithm can successfully deal with a wider range of access constraints on an Ethernet network because the algorithm has more flexibility in adapting the aggregate conference message rate. RS produces conferences with higher fidelity and lower latency than the other algorithms considered. The difference between the performance of the RS algorithm and the other algorithms increases when the Ethernet is severely constrained and when the conference media units are small compared with the Ethernet MTU.

5.4.4. Access Constraint Conclusions

The experiments in this section show the effects of access constraints on video conferences. Access constraints are easy to create on existing LANs and can cause severe degradation of conference quality. Bit rate scaling can sometimes address moderate access constraints via the secondary effects of bit rate scaling on the effective packet rate, but bit rate scaling is generally unsuccessful when severe access constraints are present. Fragmentation can exacerbate access constraints by greatly increasing the packet rate on a congested hop. With moderate levels of congestion, fragmentation may make bit rate scaling somewhat more competitive with RS, but the degree to which scaling algorithms can compete with RS decreases with increasing levels of access constraints. The performance differences between RS and the other algorithms increase when the media frames are small in relation to the network MTU since this allows RS to package multiple media frames in a single network packet. Algorithms based on the transmission control framework (*e.g.*, RS) are better equipped to handle access constraints than either non-adaptive transmission schemes or schemes that manipulate only bit rates. As a result, RS delivers higher quality conferences on an access constrained network than any of the other transmission schemes considered, even if messages are fragmented during transmission.

5.5. Dynamic Capacity Constraint

This section discusses a set of experiments where the network is capacity constrained. The section complements the previous section that showed the results achieved by several transmission schemes over access constrained networks. Comparing the relative performance of the transmission schemes under each of the "pure" congestion conditions reveals some of the inherent strengths and weaknesses in the schemes. This section shows that video scaling techniques are more competitive with the Recent Success (RS) algorithm in capacity constrained networks than in access constrained networks, but RS still delivers better overall conference quality even under capacity constraints.

5.5.1. Capacity Constraints on a Token Ring Network

5.5.1.1. Generating Capacity Constraints

We use the standard TCP/IP software supplied with AIX for all routing in the access constrained experiments. During the capacity constrained experiments, router *R1* (see Figure 5-2) runs a *relay* program in user space to route UDP traffic between the source's token ring segment and either the middle token ring segment or the Ethernet, depending on the experiment. This relay program allows us to control the maximum transmission rate (bits/second) through the router. The relay program can either impose no delay on relayed traffic or slow the transmission to arbitrarily low levels. The maximum transmission rate may be constant or may vary over time depending on the parameters supplied to the relay program.

When we specify constant constraints, the relay program limits the maximum bit transmission rate through the router to a specified level. When we configure the constraint to vary over time, the relay program periodically computes an artificial load indicator and adjusts the transmission rate based on the indicator. The load indicator is simply a number between 1 and 100. The indicator is completely artificial and is simply a tool to simulate congestion within the router. A high value for the load indicator implies high congestion and a low value implies low congestion. The indicator is calculated using a random number generator. The relay program uses the random numbers to determine the direction and magnitude of the change in load. We can limit the change in load at each evaluation using a set of thresholds. When the artificial load indicator exceeds some defined congestion threshold, the relay program

throttles the transmission rate to some specified rate. Using this program, we can create arbitrary capacity constraints within the routers.

Figure 5-26 shows a sample relay control file that dynamically constrains a router to 1.0 Mbits/sec (*pace 125*) whenever the synthetic internal load exceeds 50 (*threshold 50*). The initial internal load is generated using a random number generator and a seed value (*seed 666*). The artificial load indicator may increase at most 10 (*maxup 10*) or decrease at most 10 (*maxdown 10*) in a single recalculation.

```
buffer 16000
srcnet ip
dest 152.2.136.121
port 8001
pace 125
seed 666
threshold 50
maxup 10
maxdown 10
```

Figure 5-26: Sample "Relay" Control File

5.5.1.2. Test Environment

The experiments in this section show the results for conferences using the Baseline (BL), Video Scaling Only (VSO), Temporal Scaling Only (TSO), and Recent Success (RS) transmission control schemes on a three hop token ring network (Figure 5-3) with a dynamic capacity constraint. Router *R1* runs the *relay* routing program with the parameters shown in Figure 5-26. The *relay* program creates a capacity constraint whenever the artificial network load level exceeds 50. When the capacity constraint is active, *R1* limits throughput through the router to a maximum bit rate of 1.0 Mbits/second. No traffic generators are used in the experiments in this section.

5.5.1.3. Experimental Results

Figures 5-27, 5-28, 5-29, and 5-30 show the results for video conferences using the BL, VSO, TSO, and RS algorithms, respectively. Table 5-11 summarizes the results of the experiments. These experiments use the *High Bit Rate* (*HBR*) system. Figure 5-1 shows the operating points for this system. During the constrained periods, the bit rate throughput in *R1* is approximately half the aggregate bit rate of the conference. This capacity constraint is less severe than the access constraints discussed in the previous section, but still affects the quality of the audio and video streams. Under

BL, the delivered video frame rate oscillates between full and half rate video (Figure 5-27 (a)). The average video frame rate is about 26 FPS (Table 5-11). Audio is choppy with many small, but perceptible drops in audio delivery. There are 1,356 audio gaps during the conference (Table 5-11). Audio and video latency vary widely, but generally stay below the 250 millisecond guideline (parts (c) and (d)).

With VSO (Figure 5-28), audio performance is slightly improved over that with BL. There are 1,090 audio gaps and comparable audio latency (Table 5-11). The video frame delivery is also improved, with delivery rates generally between 20 and 30 frames per second with an average frame rate of about 27 FPS, but with a lower quality video encoding during the congested periods. There is less message loss with VSO than with BL, but as many as twenty percent of the total transmitted messages are still lost during congested periods. Figure 5-28 (f) shows that VSO used all three possible video encodings, but only made modest improvements in the overall quality of the delivered conference.

The TSO algorithm produces a better delivered audio stream with lower loss than either BL or VSO (Figure 5-29 (a) and (b)). There are only 382 audio gaps during the conference. TSO lowers the bit rate, as well as the message rate, of the conference by using temporal scaling on the video stream. The result of the temporal scaling is to essentially trade lower video frame delivery for better audio delivery during congested periods. This approach leads to less message loss and comparable audio and video stream latency (parts (c) and (d)). However, there is a high perceptual impact to video because of the significantly reduced video frame rate. The average delivered video frame rate is only about 18 FPS (Table 5-11).

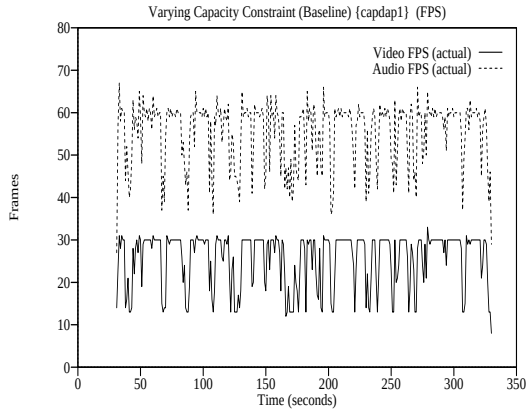
RS improves the delivered audio frame rate at much less cost to the video stream (Figure 5-30 part (a)). The conference using RS has only 146 audio gaps. The video frame rate is better than with TSO (about 25 FPS; Table 5-11). Video quality is slightly worse than with VSO, but perceptually similar. There is very little message loss (part (b)). Video latency is comparable to that experienced with the other algorithms (part (d)). Audio latency is higher with RS due to the induced latency resulting from packaging multiple audio frames into a single message. However, the maximum audio latency is below the 250 *ms* threshold and comparable to the audio latency experienced during the congested periods under the other transmission control schemes.

5.5.1.4. Conclusions about Capacity Constraints on Token Rings

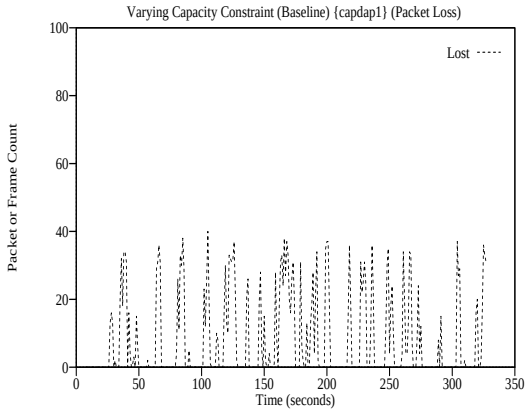
The results of the previous experiments show that capacity constraints can sometimes be successfully addressed by spatial and temporal scaling strategies. Both of these strategies reduce the bit rate of the conference and depending on the degree of the capacity constraint and the potential operating points of the conferencing system, both strategies may sufficiently lower the bit rate of the video stream to accommodate the capacity constraint.

Even though VSO improved the quality of the conference when the network was congested, it is surprising that the VSO algorithm did not have a more significant impact on the delivered stream of the conference. We speculate the reason for the disappointing performance of VSO is that even though the network is capacity constrained at *R1*, competition for buffers in the processing queue at *R1* creates a secondary access constraint generated as a result of the capacity constraint. It appears that on the routers used in this experiment buffering of incoming packets is done with a fixed set of constant size buffers and that limiting the demand for these buffers can improve the effective frame delivery rates and limit packet loss. For this reason, the TSO algorithm was more successful at delivering the audio stream than VSO even though VSO sometimes cut the video bit rate to a quarter its maximum rate, while the TSO algorithm generally never reduced the video bit rate below a third the maximum bit rate.

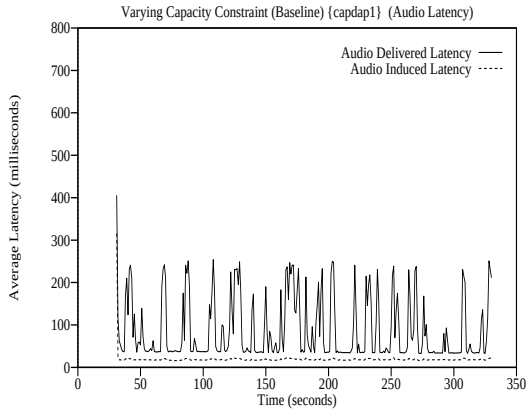
The RS algorithm has equivalent or better performance than any of the scaling algorithms. RS can make the same bit rate changes as VSO and TSO and may augment these reductions with changes to the packaging of audio to account for the secondary access constraints generated in the router buffer pool. RS produces better audio delivery rates than any other algorithm. Video frame delivery varies with the degree of congestion in the network. TSO produces audio rates only slightly lower than RS, but with inferior video quality. RS consistently produces the lowest message loss and lowest video latency. The audio latency under RS is higher on average than with the other schemes because of the induced latency associated with packaging, but RS controls audio latency and trades additional latency for improved audio fidelity (*i.e.*, fewer audio gaps).



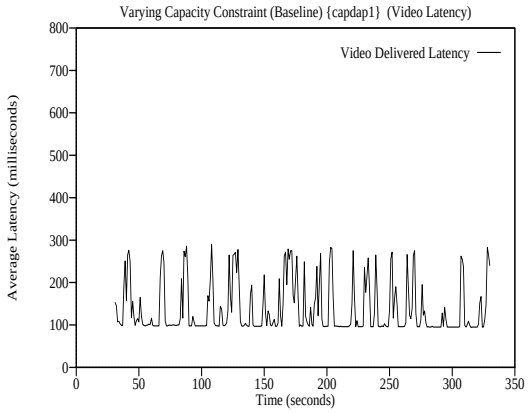
(a) Frames Per Second (FPS)



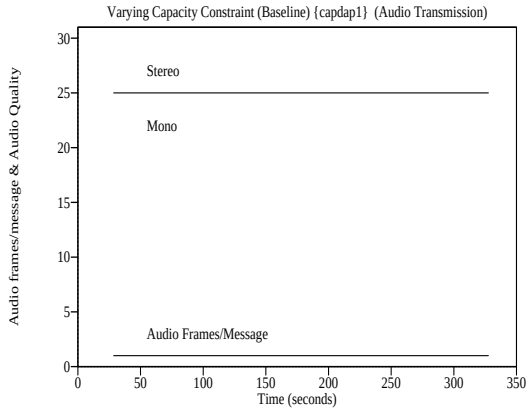
(b) Message Loss



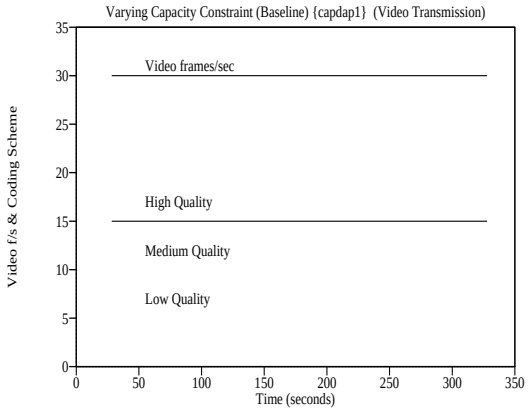
(c) Audio Latency



(d) Video Latency

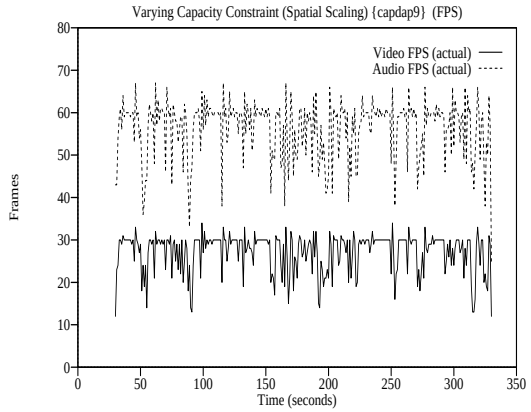


(e) Audio Transmission

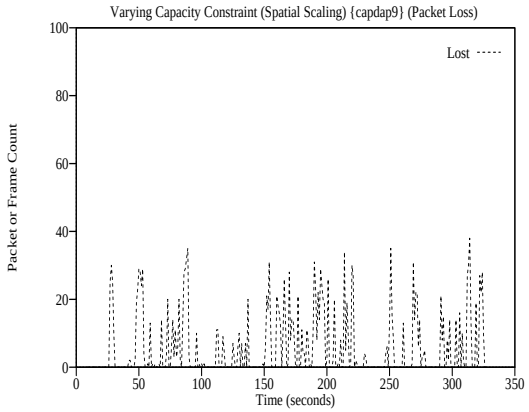


(f) Video Transmission

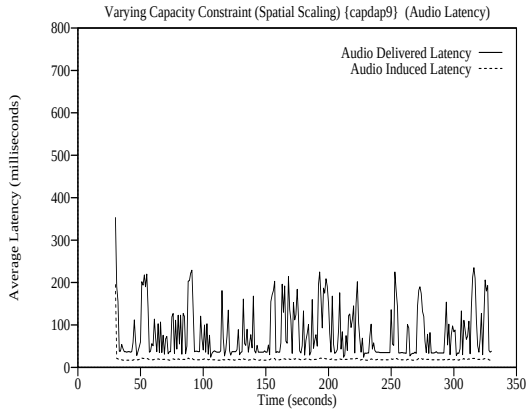
Figure 5-27: Capacity Constraint (TR-TR-TR) Experiment - Baseline



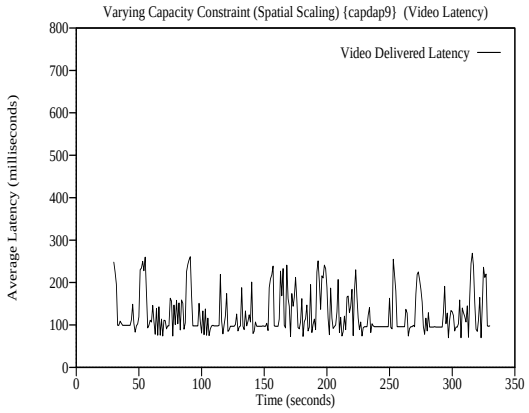
(a) Frames Per Second (FPS)



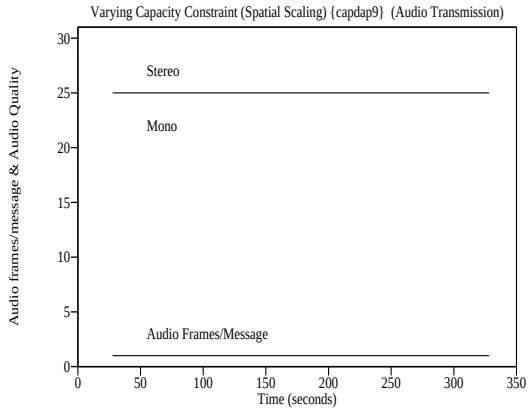
(b) Message Loss



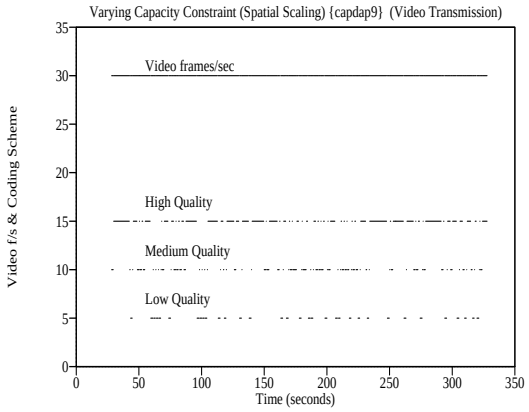
(c) Audio Latency



(d) Video Latency

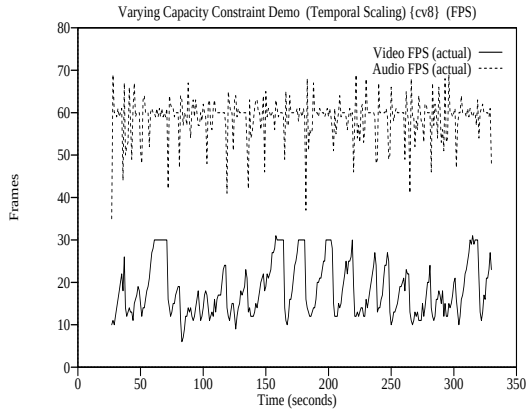


(e) Audio Transmission

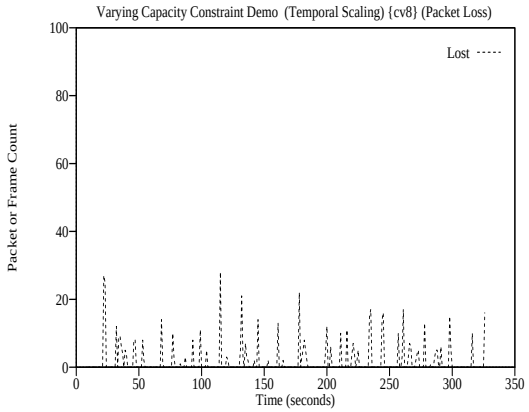


(f) Video Transmission

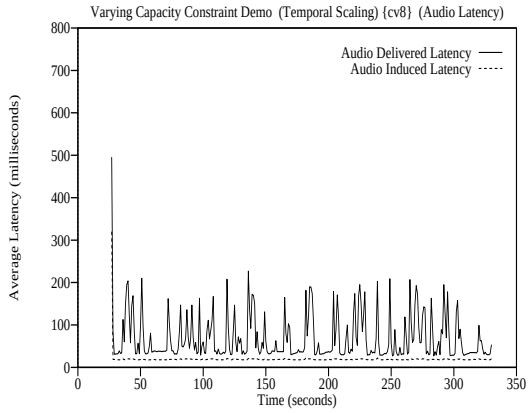
Figure 5-28: Capacity Constraint (TR-TR-TR) Experiment - Video Scaling Only



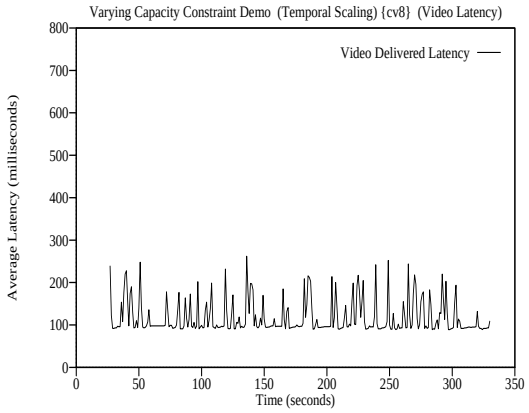
(a) Frames Per Second (FPS)



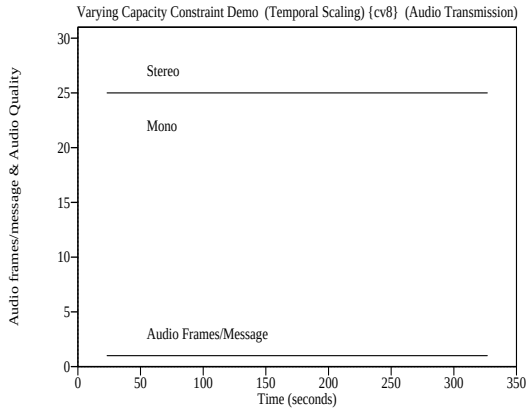
(b) Message Loss



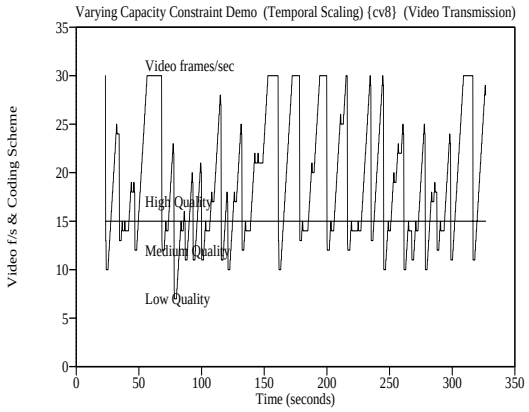
(c) Audio Latency



(d) Video Latency

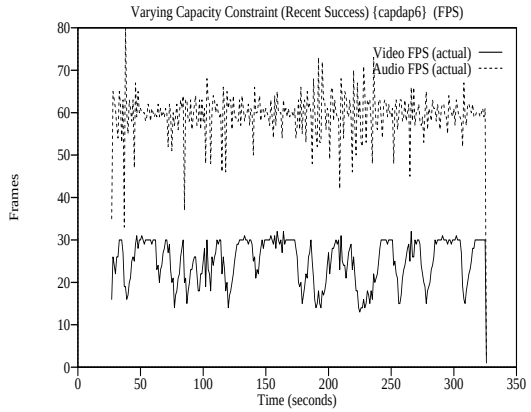


(e) Audio Transmission

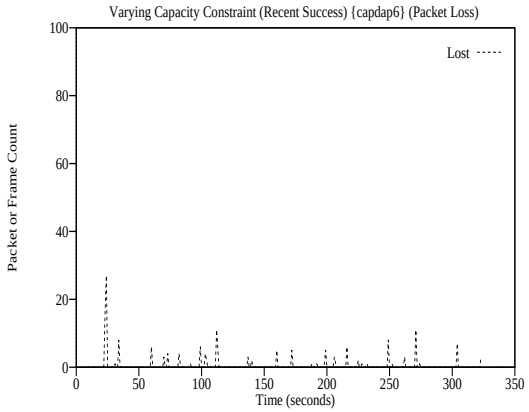


(f) Video Transmission

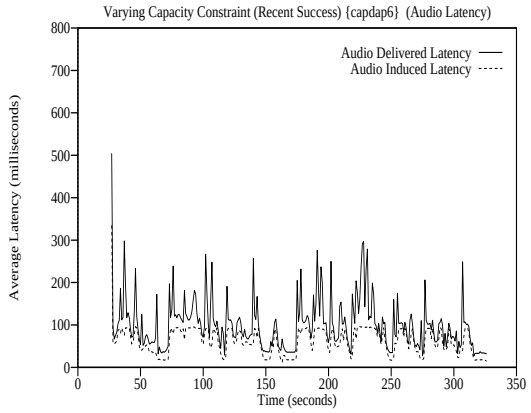
Figure 5-29: Capacity Constraint (TR-TR-TR) Experiment - Temporal Scaling Only



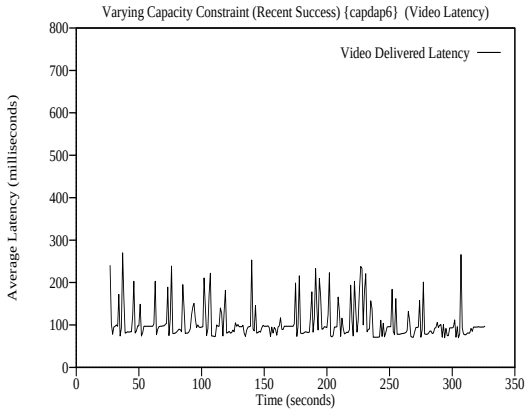
(a) Frames Per Second (FPS)



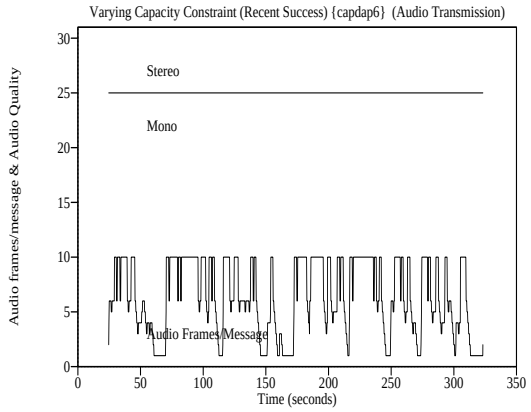
(b) Message Loss



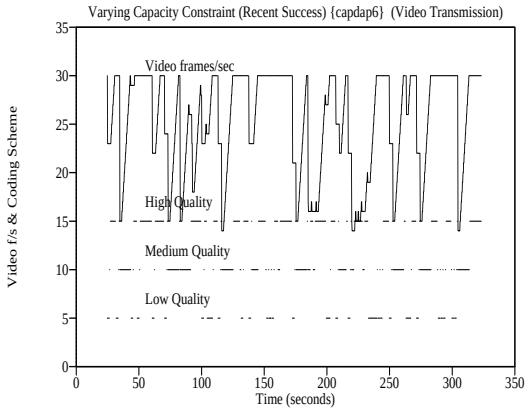
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-30: Capacity Constraint (TR-TR-TR) Experiment - Recent Success

Varying Capacity	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	55.46	56.35	58.85	59.48
Standard deviation	7.44	6.89	4.81	6.03
Minimum	29	25	37	2
Maximum	67	67	69	81
Mode/Median	60/59	60/59	60/60	60/60
Gaps	1356	1090	382	146
Audio Latency: Delivered (Induced)				
Mean	87.27 (18.47)	79.61 (18.82)	67.50 (18.81)	98.73 (61.72)
Standard deviation	72.29 (1.54)	57.62 (1.25)	49.80 (0.89)	55.61 (28.02)
Minimum	33 (16)	24 (16)	27 (17)	27 (12)
Maximum	254 (23)	235 (22)	227 (22)	298 (96)
Mode/Median	35/42 (18/18)	36/51 (18/19)	32/39 (18/19)	104/94 (92/60)
Intervals > 250 ms	3	0	0	8
Audio Messages				
Frames Sent	18017	18034	18266	17989
Msgs(frames) Lost	1340 (1340)	1066 (1066)	365 (365)	70 (115)
Mean Frames Lost	4.45	3.54	1.20	0.38
Max Frames Lost	23	24	16	19
Video FPS				
Mean	25.59	27.21	18.22	24.94
Standard deviation	6.41	4.54	6.25	5.34
Minimum	8	12	6	1
Maximum	33	34	31	32
Mode/Median	30/30	30/29	12/16	30/26
Gaps	1324	845	3596	1507
Video Latency: Delivered				
Mean	135.71	126.61	117.10	104.26
Standard deviation	60.75	48.88	38.85	39.26
Minimum	94	70	89	70
Maximum	290	269	262	270
Mode/Median	98/101	96/101	94/97	97/94
Intervals > 250 ms	37	7	2	3
Video Messages				
Frames Sent	9009	9016	5783	7610
Frames Lost	1314	832	236	111
Mean Frames Lost	4.37	2.76	0.77	0.37
Max Frames Lost	19	17	14	12

Table 5-11: Varying Capacity Constraint Experiment Summary

5.5.2. Capacity Constraints on a Network with Small MTU

5.5.2.1. Experimental Environment

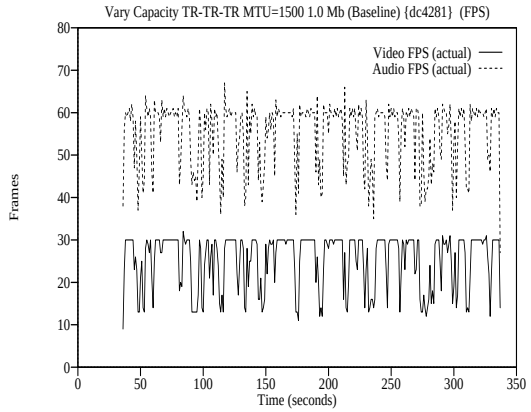
The experiments in this section are identical to the previous section except that the maximum transmission unit on the middle token ring in Figure 5-3 is set to 1,500 bytes and thus messages transmitted to the destination by *R1* must be fragmented.

5.5.2.2. Experimental Results

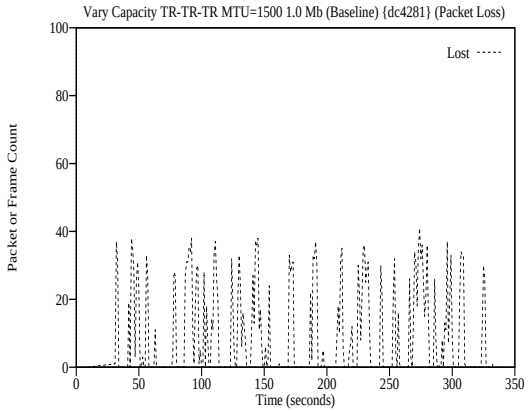
Figures 5-31, 5-32, 5-33, and 5-34 show the results of experiments using the BL, VSO, TSO, and RS algorithms, respectively. Table 5-12 summarizes the results. These experiments use the HBR system (Figure 5-1). The results are virtually identical to the results in the previous section when the maximum transmission unit was 17,800 bytes per packet. RS again provides the best overall performance with slightly inferior video when compared with VSO, but superior audio quality and much less loss.

5.5.2.3. Capacity Constraints with Small MTU Conclusions

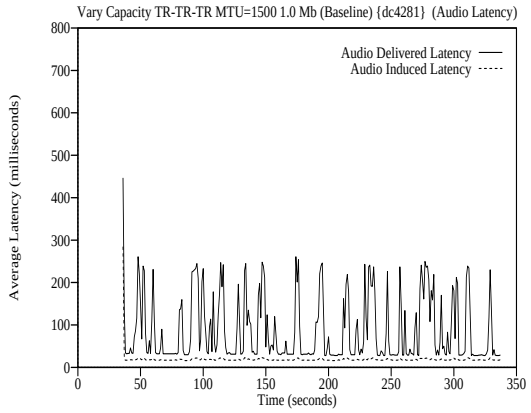
The experiments in this section show that changing only the maximum transmission unit of a network downstream from a capacity constraint has little or no effect on the quality of the delivered conference. This result is in contrast with the results on an access constrained network, where reducing the maximum transmission unit caused significant degradation of conference quality for all transmission schemes. The increased demand for access to the network caused by fragmentation exacerbates the effects of the access constraint. Since capacity constraints are caused by excessive bit rates, changes in packet rates due to fragmentation have little effect on the delivered conference quality. Replacing the middle token ring in Figure 5-3 (*i.e.*, as in Figure 5-14) with an unloaded Ethernet, which also has an MTU of 1500 bytes, produces results similar to those shown in Figures 5-31, 5-32, 5-33, and 5-34 and are not shown here. This result is not surpising since the Ethernet and the middle token ring are unloaded and are using the same MTU. The performance of the network differs only slightly between the two configurations since the middle token ring and the Ethernet are both acting as dedicated transmission links from *R1* to *R2*.



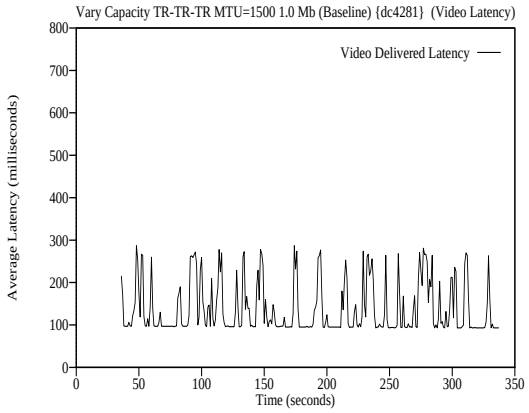
(a) Frames Per Second (FPS)



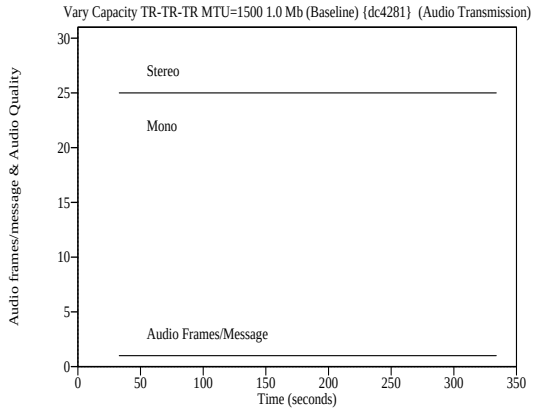
(b) Message Loss



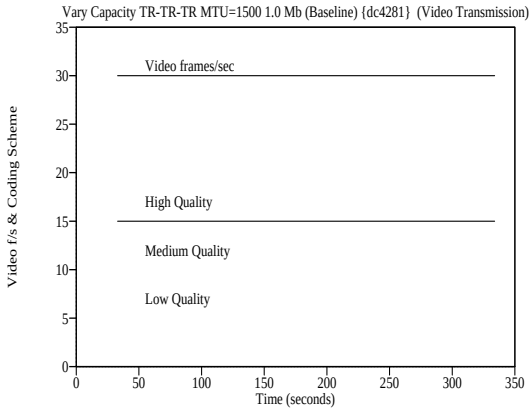
(c) Audio Latency



(d) Video Latency

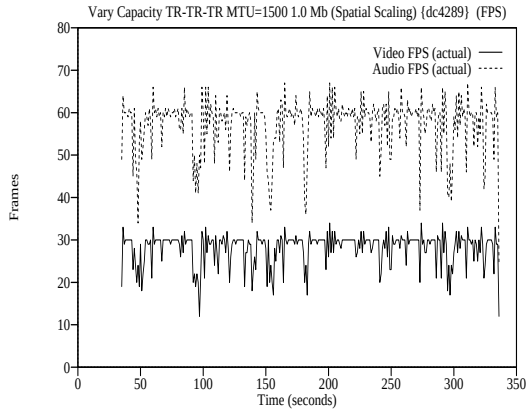


(e) Audio Transmission

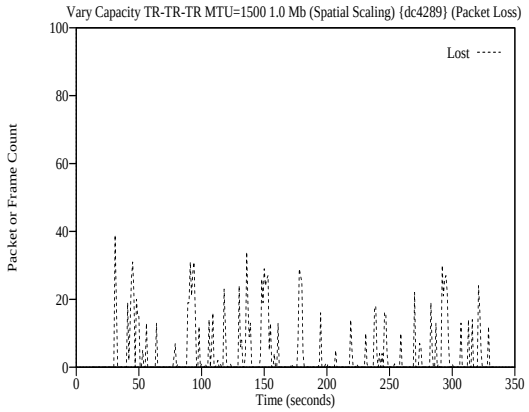


(f) Video Transmission

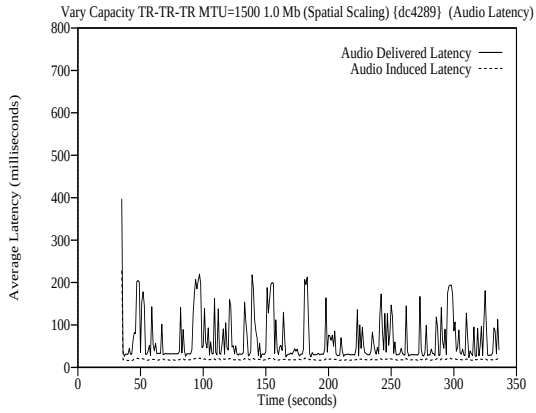
Figure 5-31: Capacity Constraint (MTU=1,500) Experiment - Baseline



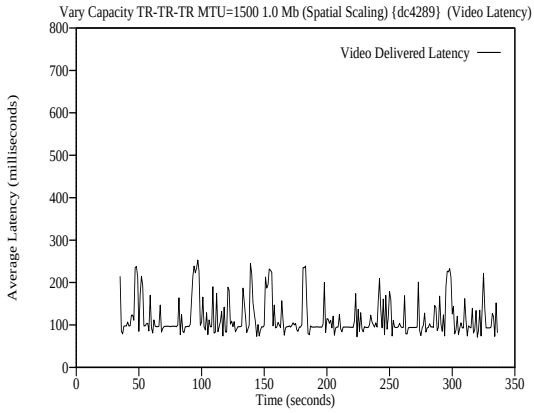
(a) Frames Per Second (FPS)



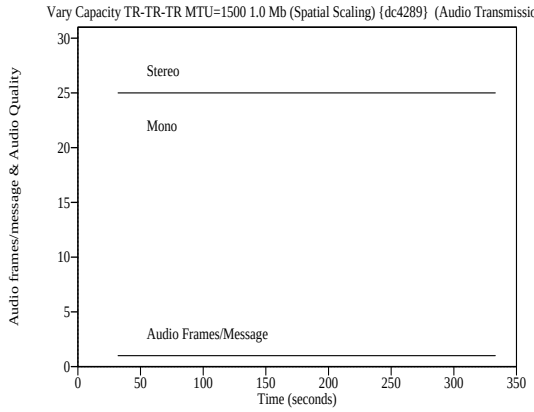
(b) Message Loss



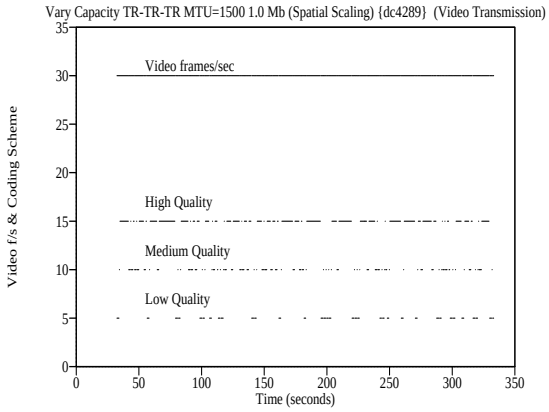
(c) Audio Latency



(d) Video Latency

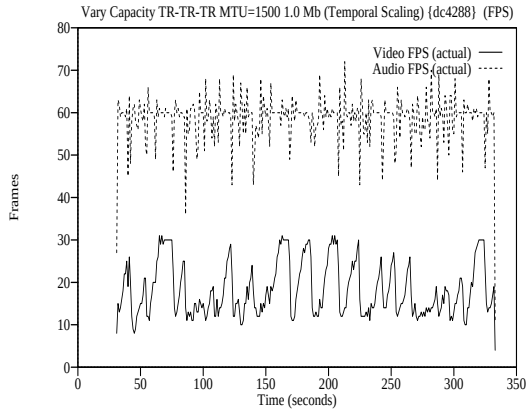


(e) Audio Transmission

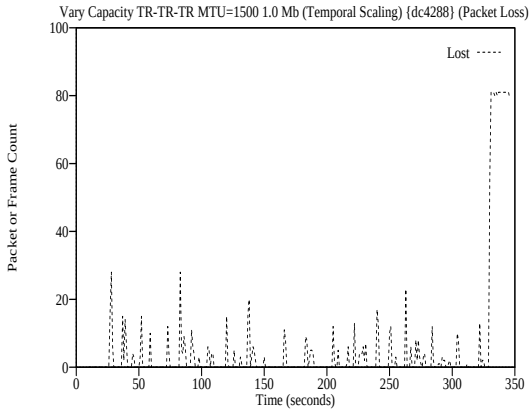


(f) Video Transmission

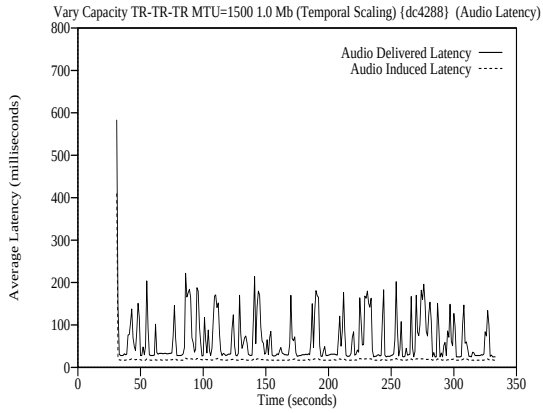
Figure 5-32: Capacity Constraint (MTU=1,500) Experiment - Video Scaling Only



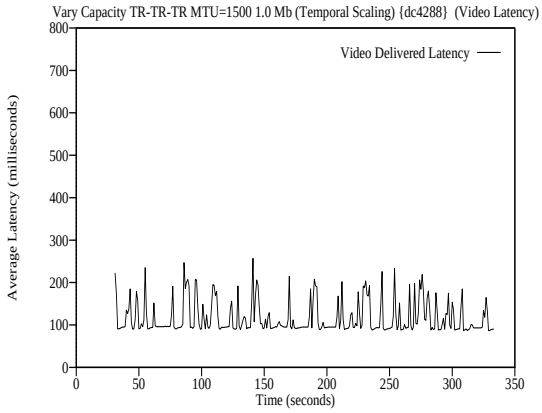
(a) Frames Per Second (FPS)



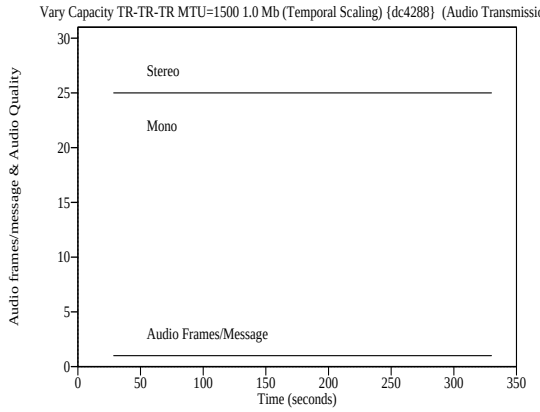
(b) Message Loss



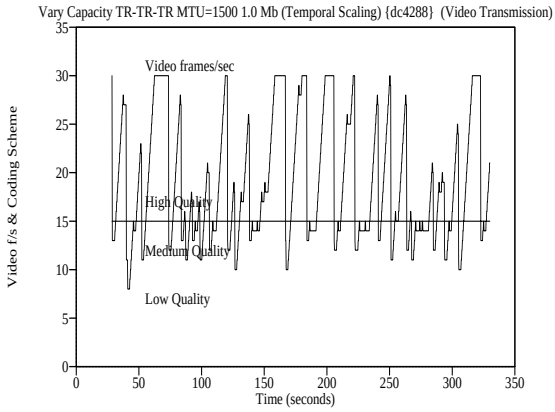
(c) Audio Latency



(d) Video Latency

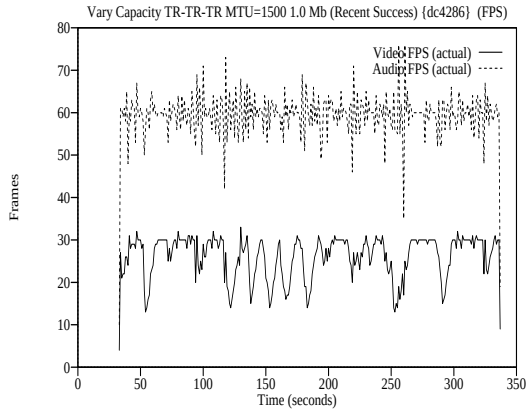


(e) Audio Transmission

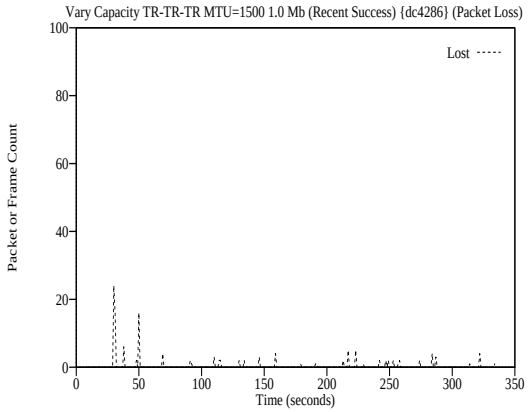


(f) Video Transmission

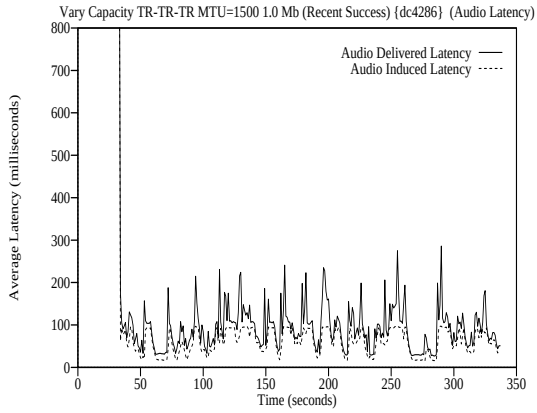
Figure 5-33: Capacity Constraint (MTU=1,500) Experiment - Temporal Scaling Only



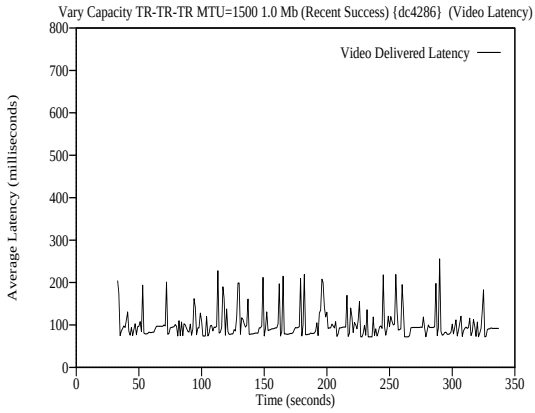
(a) Frames Per Second (FPS)



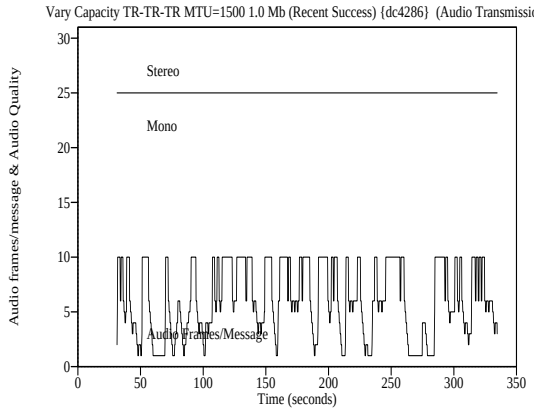
(b) Message Loss



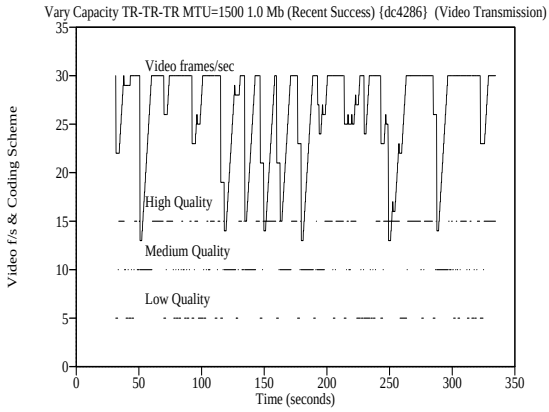
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-34: Capacity Constraint (MTU=1,500) Experiment - Recent Success

Capacity MTU=1500	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	55.25	57.22	58.69	59.55
Standard deviation	7.65	6.65	5.53	5.06
Minimum	27	24	11	19
Maximum	67	67	72	76
Mode/Median	60/69	60/60	60/60	60/60
Gaps	1434	842	386	136
Audio Latency: Delivered (Induced)				
Mean	86.62 (18.40)	65.46 (18.65)	65.12 (18.18)	92.07 (61.18)
Standard deviation	75.36 (1.84)	52.67 (1.28)	50.83 (1.19)	48.47 (27.84)
Minimum	28 (16)	23 (16)	25 (16)	27 (15)
Maximum	261 (23)	220 (22)	222 (22)	286 (98)
Mode/Median	31/39 (17/18)	30/37 (18/18)	28/35 (17/18)	30/88 (93/60)
Intervals > 250 ms	2	0	0	2
Audio Messages				
Frames Sent	18092	18109	19177	18259
Msgs(frames) Lost	1410 (1410)	815 (815)	1356 (1356)	45 (104)
Mean Frames Lost	4.67	2.69	4.24	0.34
Max Frames Lost	23	24	60	12
Video FPS				
Mean	25.28	28.14	18.27	26.05
Standard deviation	6.40	3.80	6.30	4.94
Minimum	11	12	4	9
Maximum	32	34	31	33
Mode/Median	30/30	30/30	12/16	30/28
Gaps	1435	573	3546	1200
Video Latency: Delivered				
Mean	138.14	116.41	116.43	100.20
Standard deviation	61.22	43.01	38.26	33.22
Minimum	93	71	87	72
Maximum	287	253	257	256
Mode/Median	95/100	95/97	95/96	
Intervals > 250 ms	36	1	1	1
Video Messages				
Frames Sent	9046	9055	6153	8027
Frames Lost	1422	559	599	86
Mean Frames Lost	4.71	1.84	1.87	0.28
Max Frames Lost	19	18	21	12

Table 5-12: Varying Capacity Constraint with MTU=1500 Experiment Summary

5.5.3. Capacity Constraint Conclusions

Structural capacity constraints are perhaps the easiest constraint to visualize since the constraint corresponds to the physical capacity of some network element. It is easy to see that it is fundamentally impossible to transmit a conference at a greater bit rate than the minimum physical bit rate capacity. Due to the high bit rates associated with video conferencing, it is easy to create examples of structural capacity constraints. For example, a T1 communications line, which has a maximum bit rate of 1.544 Mbits/second, has inadequate capacity to carry the full 2 Mbit/second maximum bit rate of the conferencing system used in the experiments in this section. Congestion capacity constraints, on the other hand, are harder to create. The capacity of a telecommunications link between two nodes does not change over time; only the competition for control of the resource may change. Similarly, the internal capacity of a router system bus or the processing rate of the physical CPU does not change. The effective capacity or processing rate in the router may change in the sense that processor and bus sharing may reduce the share of the resource dedicated to a particular conference media stream, but given current router technology, congestion capacity constraints are relatively difficult to create. Access constraints, on the other hand, are easy to create, both in a primary constraint (*e.g.,* due to competition for a shared medium) and as a secondary constraint (*e.g.,* due to buffer competition resulting from some primary constraint).

Addressing a capacity constraint fundamentally implies lowering the bit rate of the streams, but the secondary access constraints sometimes resulting from the primary capacity constraint may also make reducing the packet rate desirable. The experiments in this section show that scaling the streams improves the delivered quality of the conference when compared with the non-adaptive transmission scheme. They also show that even better results can be achieved when scaling is combined with packaging such that the primary capacity constraint and the secondary access constraint are addressed simultaneously. Scaling techniques are much more competitive with the Recent Success algorithm for capacity constraints when compared with access constraints because the fundamental strategy of scaling is bit rate reduction, which directly addresses the primary capacity constraint. Nevertheless, the Recent Success algorithm still produces better overall conference quality by using a combination of scaling and packaging. The Video Scaling Only algorithm produces better video quality than Recent Success, but Recent Success has better audio quality

and much lower loss while still providing good quality video under low congestion and adequate quality during congested periods.

5.6. Combination Constraints

The experiments in this section combine the access and capacity constraints in the earlier sections to produce network conditions with both constraints simultaneously. As with the other experiments, the delivered conference quality under these conditions is measured for a set of candidate transmission schemes on several network topologies. The primary intent of this section is to demonstrate that the effects of combination constraints may be controlled by adaptive transmission control techniques.

5.6.1. Combination Constraints on Token Ring Networks

5.6.1.1. Experimental Environment

The experiment described in this section uses the network in Figure 5-3. In this experiment, all token ring segments have an MTU of 17,800 bytes. The router *R1* runs the *relay* program to perform routing. The *relay* program uses the parameters shown in Figure 5-26, which results in a capacity constraint in *R1* that varies in degree over time. Traffic generators are attached to the middle network and produce an access constraint on the middle network that also varies in degree over time. The parameters for the traffic generators are shown in Figure 5-3. In this environment, *R1* sometimes suffers under a capacity constraint within the router, sometimes experiences a severe access constraint trying to access the middle token ring, and sometimes experiences both constraints simultaneously.

5.6.1.2. Experimental Results

Figures 5-35, 5-36, 5-37, and 5-38 show the results for conferences using the BL, VSO, TSO, and algorithms, respectively. Table 5-13 summarizes the results. Figure 5-35 (a) shows that audio is poor when using the BL algorithm. There are 3,510 audio gaps during the conference (Table 5-13). There are long periods with low audio delivery rates. Video is delivered with an average frame rate of 23 FPS, both both audio and video experience many periods with extremely high latencies (700-800 *ms* for audio and over 800 *ms* for video; see parts (c) and (d) and Table 5-13).

Approximately a quarter of all measurement intervals have latencies higher than the 250 *ms* guideline. Overall, the perceptual quality of the conference is very poor. Furthermore, the conference also has poor quality from a network perspective since message loss is high (part (b)).

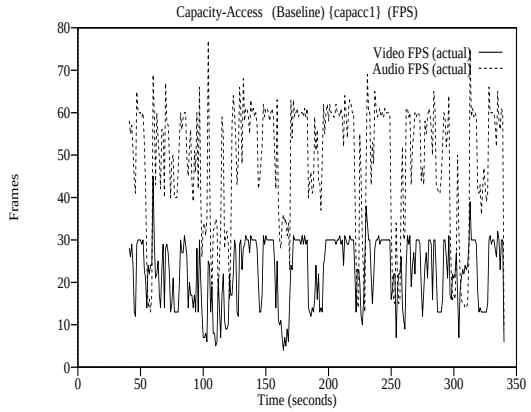
Figure 5-36 shows that VSO does not perform much better than BL in this environment. Video frame delivery is slightly improved with an average delivered frame rate of 24 FPS (Table 5-13), but the frames are often coded using the lowest quality encoding (part (f)). There are many periods with low audio delivery rates (part (a)) and there are 3,426 audio gaps (Table 5-13). Both streams still experience intervals with very high latency (700-800 *ms*; parts (c) and (d)) and approximately a quarter of the intervals have latencies above 250 *ms*. Message loss remains high (part (b)).

From a network perspective, TSO does a better job of adapting to the network conditions, as shown by the reduced message loss in Figure 5-37 (b). Audio delivery is slightly improved, but still poor. There are 2,181 audio gaps during the conference (Table 5-13). Video delivery frame rates are much lower than with BL or VSO. TSO only delivers an average of 12 video frames per second (Table 5-13). Latency for both streams remains poor (parts (c) and (d) and Table 5-13).

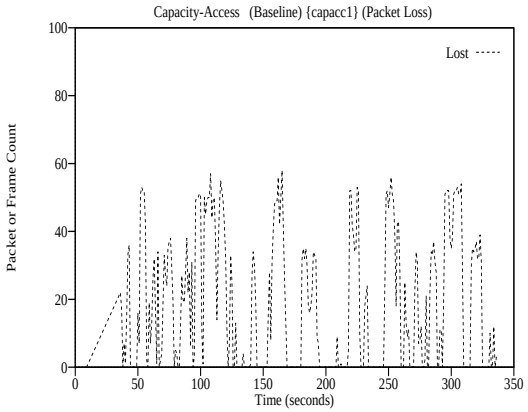
RS dramatically outperforms the other adaptation schemes in this environment. Figure 5-38 (a) shows that audio fidelity is much better with RS (*e.g.*, there are only 147 audio gaps; Table 5-13). Video delivery generally varies between half- and full-rate NTSC video, depending on the level of congestion. The average delivered video frame rate is 24 FPS. This rate is better than with TSO and comparable to that with BL and VSO. The average audio and video latencies are less than half that delivered by the other schemes. Less than five percent of the intervals exceed the 250 *ms* guideline (parts (c) and (d) and Table 5-13). There is almost no message loss (part (b)). Parts (e) and (f) shows that RS achieves these results by frequently adjusting the packaging of audio frames, the generated frame rate of video, and the quality of the video coding scheme. In other words, RS uses the combined power of bit rate and message rate adjustments to adapt to the conditions present in the network. Even under the extreme network conditions present in this experiment, RS delivers a quality conference, while the other candidate transmission schemes cannot.

5.6.1.3. Conclusions for Combination Constraints on Token Ring Networks

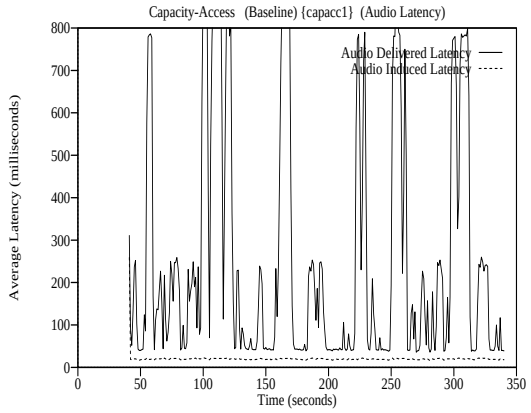
RS far outperforms the other transmission schemes when there is a combination of severe capacity and access constraints. The success of RS is due to its foundation in the transmission control framework. RS is able to apply spatial and temporal video scaling together with adaptive packaging of audio frames to address both access and capacity constraints. Algorithms using only a subset of these adaptations (*e.g.*, VSO and TSO) are not able to adapt to the types and degrees of constraints that may be present in the network. On the other hand, the transmission control framework is not a panacea. There are clearly levels of congestion that RS either cannot address or can only partially address. For example, in the experiments in this section, RS often delivers only a portion of the video stream and sometimes violates the latency guidelines. Nevertheless, RS can produce a good quality conference when other techniques cannot.



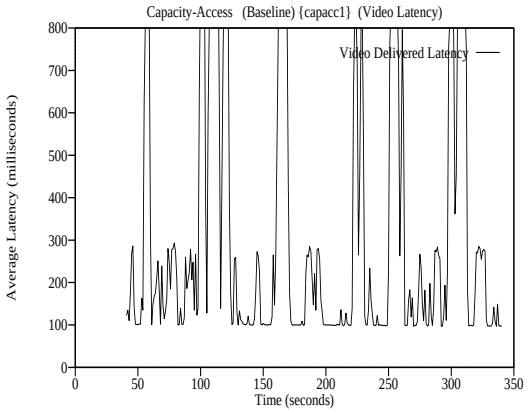
(a) Frames Per Second (FPS)



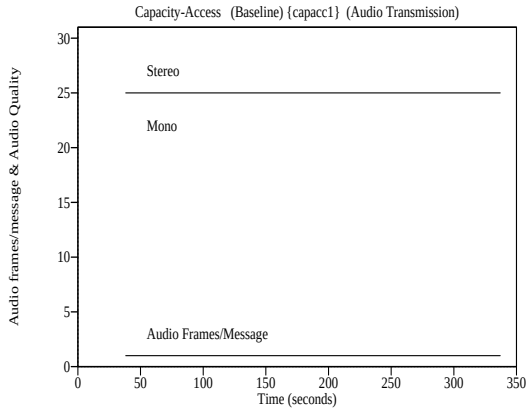
(b) Message Loss



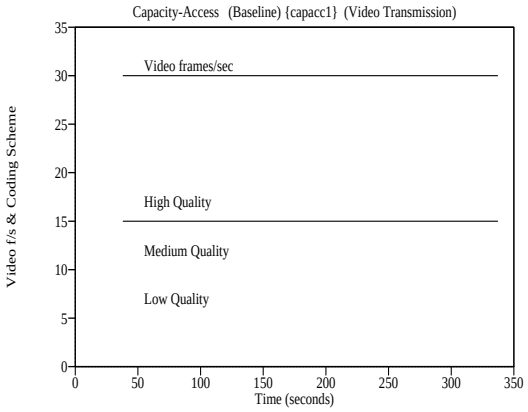
(c) Audio Latency



(d) Video Latency

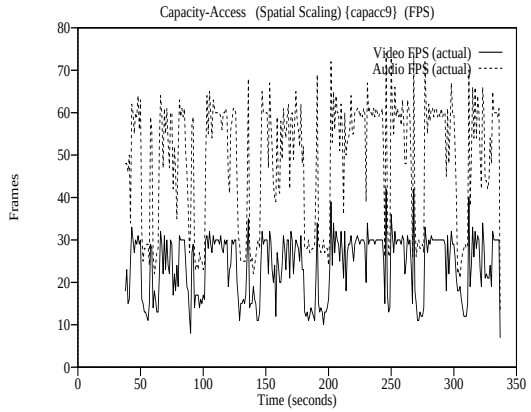


(e) Audio Transmission

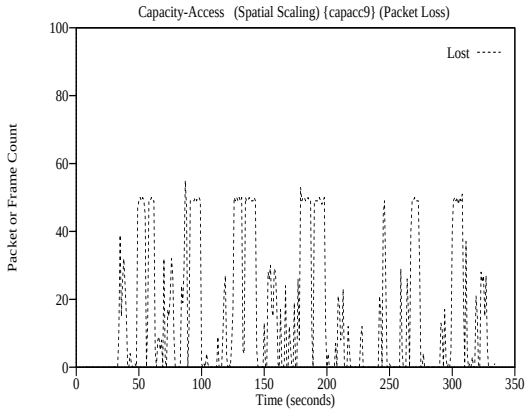


(f) Video Transmission

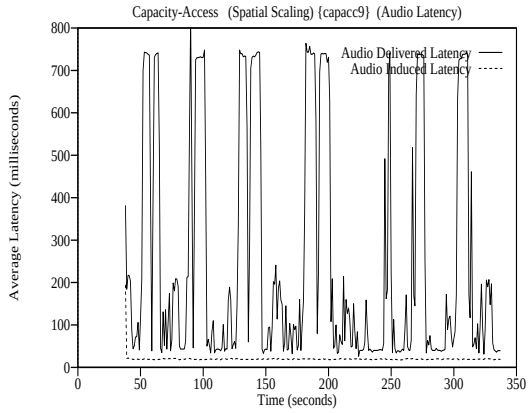
Figure 5-35: Combination Constraint (TR-TR-TR) Experiment - Baseline



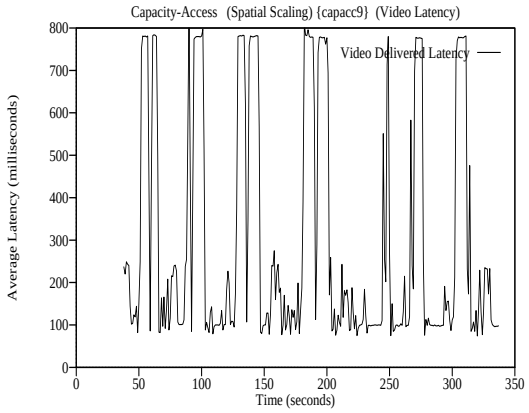
(a) Frames Per Second (FPS)



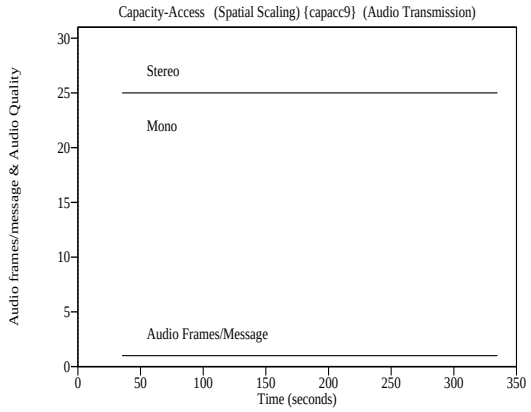
(b) Message Loss



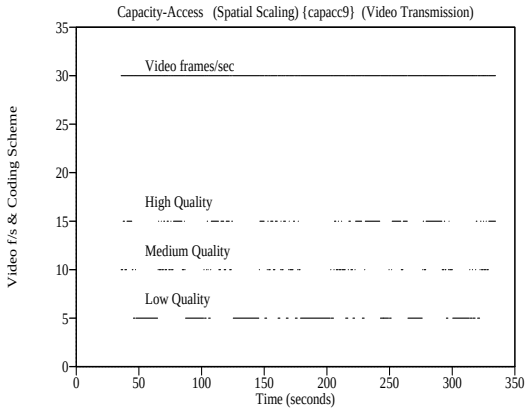
(c) Audio Latency



(d) Video Latency

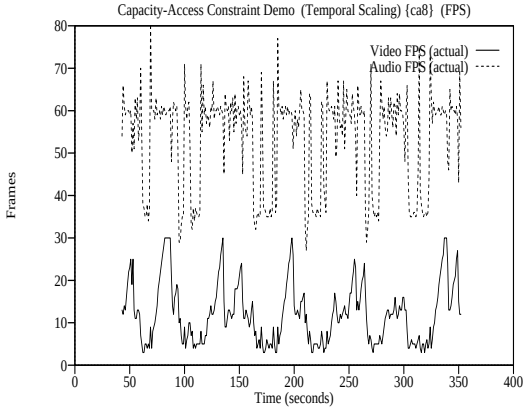


(e) Audio Transmission

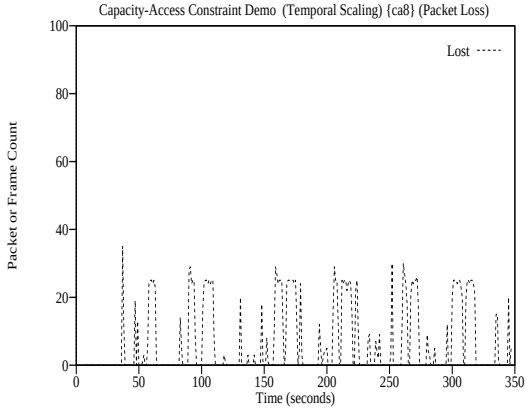


(f) Video Transmission

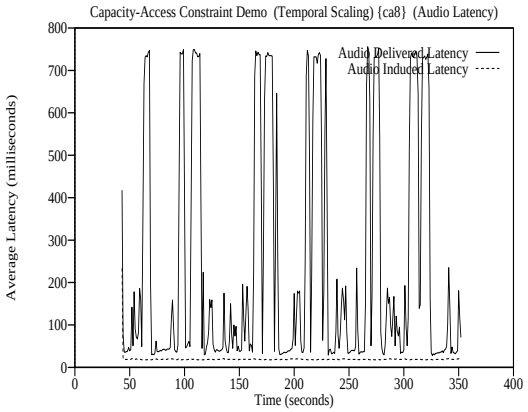
Figure 5-36: Combination Constraint (TR-TR-TR) Experiment - Video Scaling Only



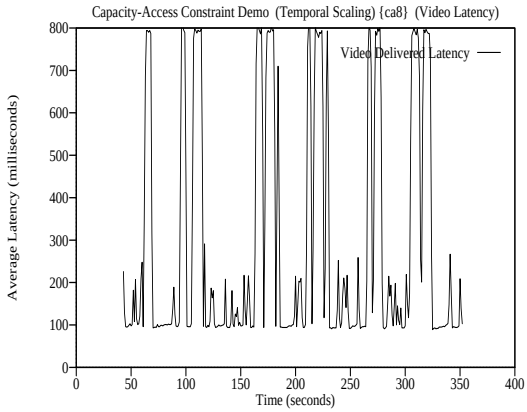
(a) Frames Per Second (FPS)



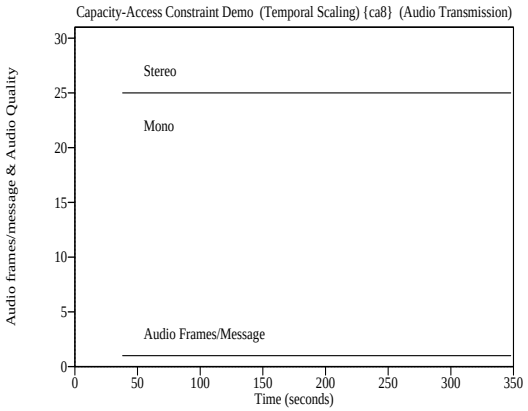
(b) Message Loss



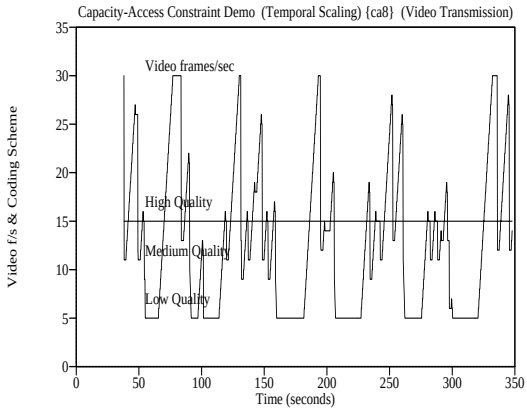
(c) Audio Latency



(d) Video Latency

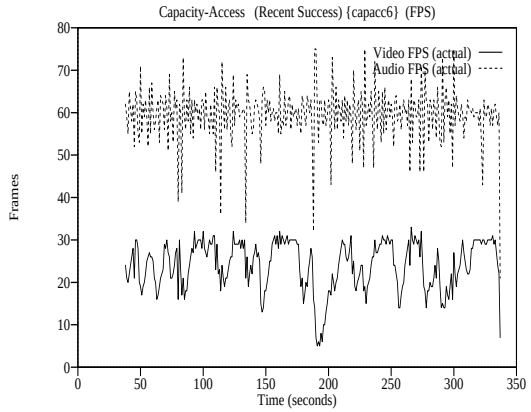


(e) Audio Transmission

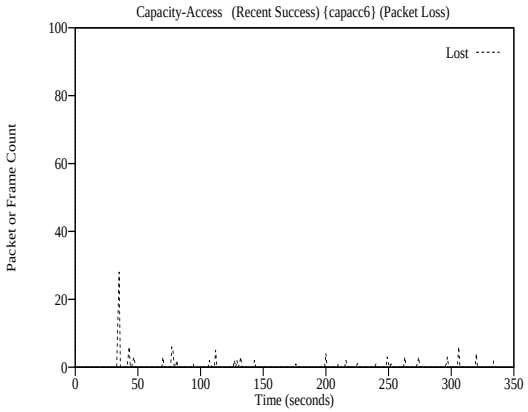


(f) Video Transmission

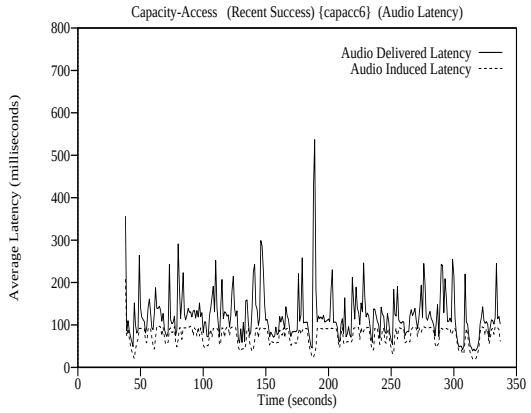
Figure 5-37: Combination Constraint (TR-TR-TR) Experiment - Temporal Scaling Only



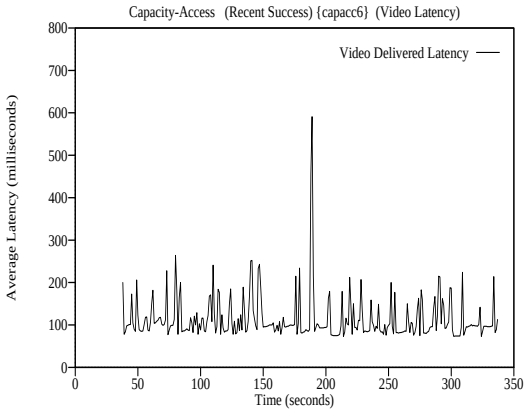
(a) Frames Per Second (FPS)



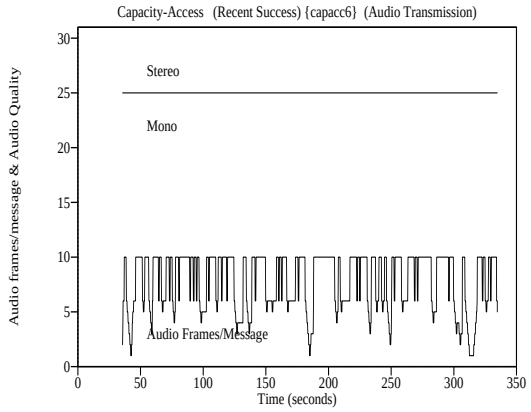
(b) Message Loss



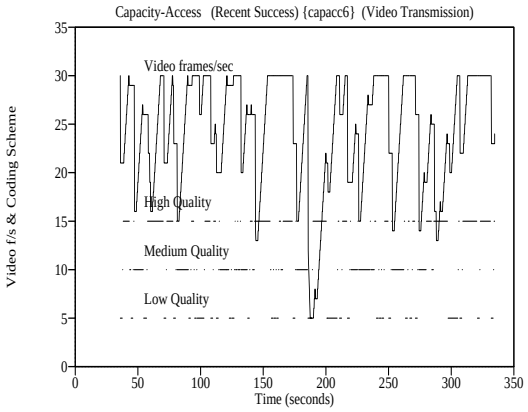
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-38: Combination Constraint (TR-TR-TR) Experiment - Recent Success

Capacity-Access	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	48.17	48.52	52.88	59.52
Standard deviation	15.01	14.85	12.04	6.42
Minimum	9	13	0	21
Maximum	77	74	82	75
Mode/Median	60/54	60/55	60/59	63/60
Gaps	3510	3426	2181	141
Audio Latency: Delivered (Induced)				
Mean	257.00 (19.72)	254.01 (19.42)	239.75 (18.89)	121.75 (74.66)
Standard deviation	288.81 (1.27)	278.03 (0.84)	277.68 (1.28)	56.92 (20.85)
Minimum	36 (16)	26 (17)	0 (0)	39 (19)
Maximum	945 (23)	821 (22)	756 (21)	537 (98)
Mode/Median	41/124 (19/20)	40/103 (19/19)	36/77 (19/19)	107/112 (92/83)
Intervals > 250 ms	78	85	87	10
Audio Messages				
Frames Sent	17982	18016	18632	18025
Msgs(frames) Lost	3479 (3479)	3403 (3403)	2154 (2154)	32 (106)
Mean Frames Lost	11.56	11.31	6.93	0.35
Max Frames Lost	48	39	29	20
Video FPS				
Mean	22.53	24.07	12.05	23.99
Standard deviation	7.98	7.36	7.13	5.78
Minimum	4	7	0	5
Maximum	45	42	30	33
Mode/Median	30/24	30/27	5/11	30/25
Gaps	2232	1780	5571	1797
Video Latency: Delivered				
Mean	305.80	298.65	293.02	113.47
Standard deviation	290.30	276.10	280.77	52.61
Minimum	97	74	0	73
Maximum	993	866	813	591
Mode/Median	100/158	99/138	94/118	97/97
Intervals > 250 ms	103	92	99	5
Video Messages				
Frames Sent	8990	9008	3937	7320
Frames Lost	2208	1768	182	96
Mean Frames Lost	7.36	5.87	0.59	0.32
Max Frames Lost	27	25	18	14

Table 5-13: Capacity-Access Constraint TR-TR-TR Experiment Summary

5.6.2. Combination Constraints on Ethernet Networks

5.6.2.1. Experimental Environment

This section describes three experiments using the BL, VSO, TSO, and RS control schemes across the network shown in Figure 5-14. The three experiments all have a combination of capacity and access constraints, but with different degrees of access constraint on the Ethernet network. All three experiments use the HBR video conferencing system described by the operating points in Figure 5-1. In all three experiments, the capacity constraint is generated by running the *relay* routing program on router *R1* using the parameters specified in Figure 5-26. These parameters cause a varying capacity constraint that sometimes limits the bit rate through the router to 1.0 Mbits/second. Three different traffic generation scripts (see Table 5-6) are used for the experiments in this section. The first experiment uses Access Load 2 (see Table 5-7) where six traffic generators are used on the Ethernet network. This leads to a varying load on the Ethernet with a moderate to heavy access constraint. The second experiment uses Access Load 3, which is the same as Access Load 2, but adds a seventh traffic generator using script 1 to make the access constraint more severe than with load 1. The last experiment uses Access Load 4, which has the same traffic generators as Access Load 3, but with one of the generators from load 3 becoming more active, producing an even more severe access constraint.

5.6.2.2. Ethernet Combination Constraint - Experiment 1 Results

Figure 5-39, 5-40, 5-41, and 5-42 shows the results of experiment 1 for the BL, VSO, TSO, and RS transmission control schemes, respectively. Table 5-14 summarizes the results. This experiment has a varying capacity constraint at router *R1* (see Figure 5-14) and has a varying access constraint on the Ethernet generated by the Access Load 2 configuration (see Table 5-7).

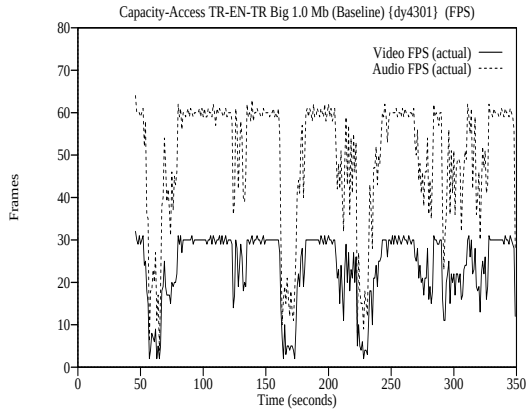
Figure 5-39 shows that in this environment the BL algorithm experiences many periods of poor conference quality. During periods with heavy congestion (*e.g.*, between 200 and 250 seconds into the conference), audio and video frame delivery rates are low (part (a)), message loss is high (part (c)), and stream latencies exceed the 250 *ms* guideline (parts (c) and (d)). There are 3,125 audio gaps and the average delivered video frame rate is 24 FPS (Table 5-14).

Figure 5-40 shows VSO does a better job than BL. There are periods when audio quality deteriorates (*e.g.*, at approximately 260 seconds into the conference in part (a)) and both streams experience periods of high latency (parts (c) and (d)), but the conference quality is significantly better than with the BL algorithm. There are 1,383 audio gaps and the average delivered video frame rate is 27 FPS. Many messages are still lost in transit (part (b)), but fewer than with BL. VSO achieves these results by using the low quality video encoding during the worst periods of congestion (part (f)), which reduces the bit and, indirectly, the message rates of the video stream.

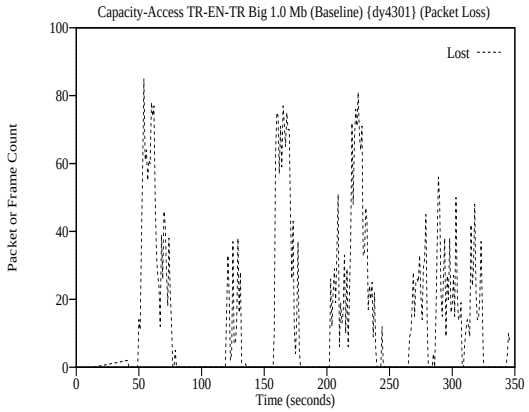
TSO (Figure 5-41) does not deliver the same video frame rate as VSO (the average frame rate is only 12 FPS; Table 5-14), but audio frame delivery is improved (*e.g.*, there are 625 audio gaps compared with 1,383 with VSO). Message loss is reduced (part (b)). Stream latencies are improved at the expense of the video frame rate (parts (c) and (d)). Figure 5-41 part (f) shows that the TSO algorithm frequently reduces the transmitted video frame rate to the minimum rate allowed (5 frames per second in this experiment). By reducing the video message and, indirectly, bit rates, TSO delivers a conference with superior quality to that using either BL or VSO for all criteria except for delivered video frame rate (which is much worse than with VSO).

Figure 5-42 shows that the RS algorithm matches or improves the performance of the TSO algorithm in all areas and, in particular, improves the delivered video frame rate. RS delivers high quality, full rate video during periods of low congestion and drops to low levels of video frame delivery only during periods of very high congestion. RS delivers a lower video frame rate (19 FPS) than VSO, but a much higher rate than TSO (Table 5-14). Furthermore, RS gives the best audio throughput of any of the algorithms with only 332 gaps during the conference. Due to the induced latency associated with audio packaging, RS delivers an audio stream with higher average latency than VSO or TSO, but audio latency is almost always below the 250 *ms* guideline. RS only experiences very short periods of high latency at the onset of heavy congestion.

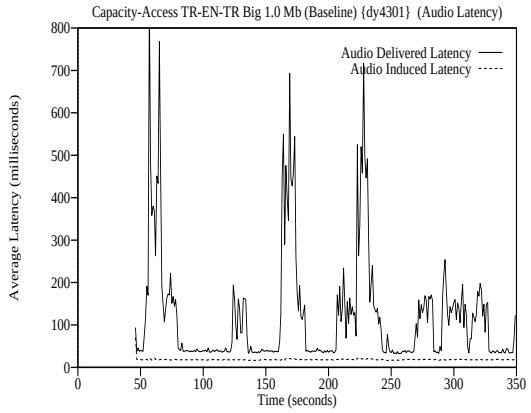
In experiment 1, RS delivers a better overall conference than any of the other algorithms; however, the performance of VSO and TSO is competitive with RS. This experiment gives an example of a network environment where a scaling-only algorithm can work almost as well as a two-dimensional adaptive algorithm. In experiments 2 and 3, the degree of network congestion increases and the experiment results show the divergence of performance among the algorithms.



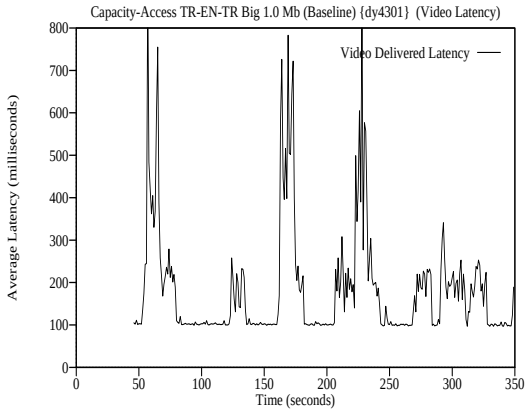
(a) Frames Per Second (FPS)



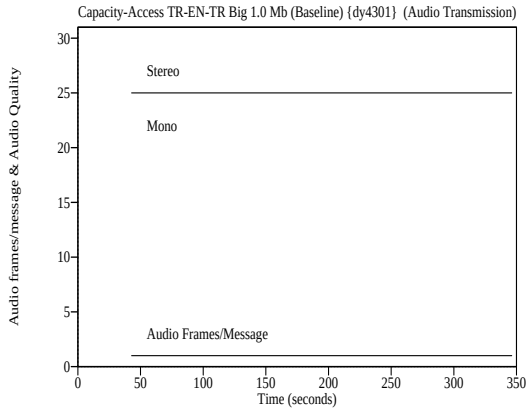
(b) Message Loss



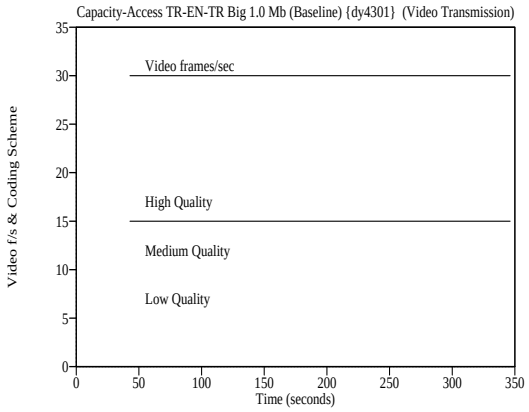
(c) Audio Latency



(d) Video Latency

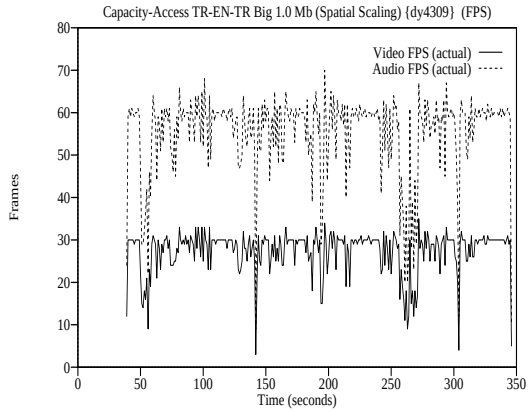


(e) Audio Transmission

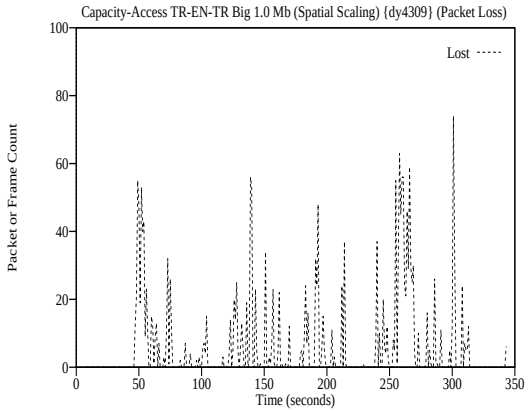


(f) Video Transmission

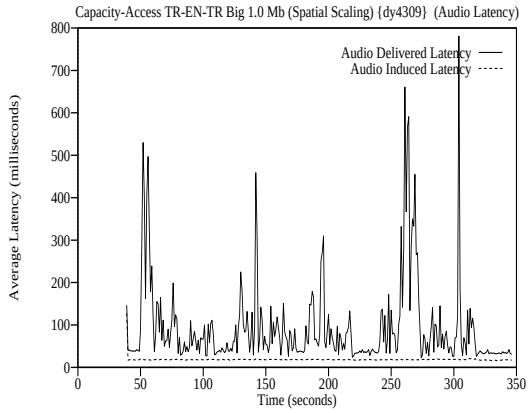
Figure 5-39: Combination Constraint (TR-EN-TR) Experiment 1 - Moderately Constrained - Baseline



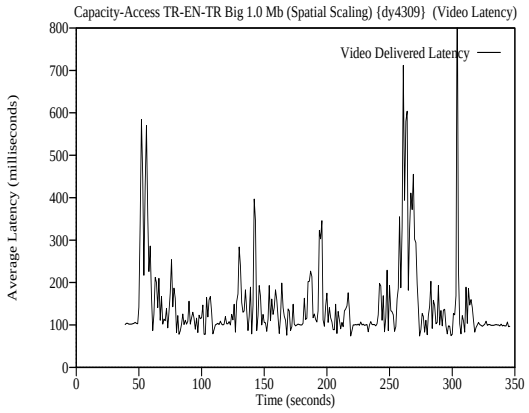
(a) Frames Per Second (FPS)



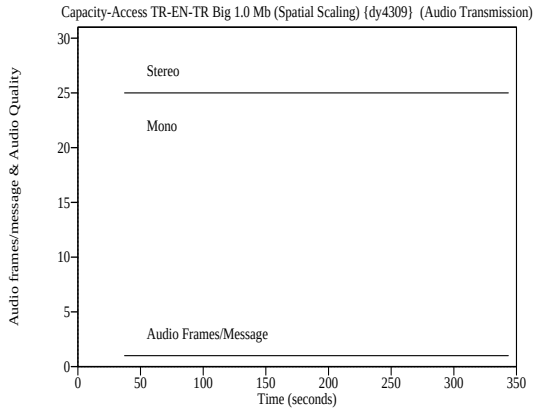
(b) Message Loss



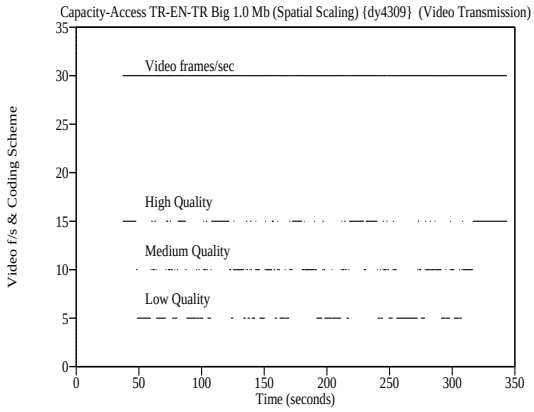
(c) Audio Latency



(d) Video Latency

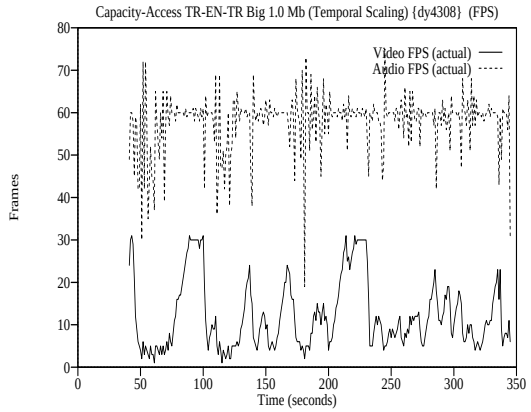


(e) Audio Transmission

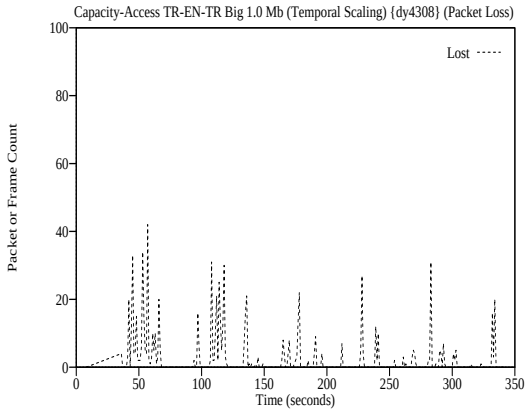


(f) Video Transmission

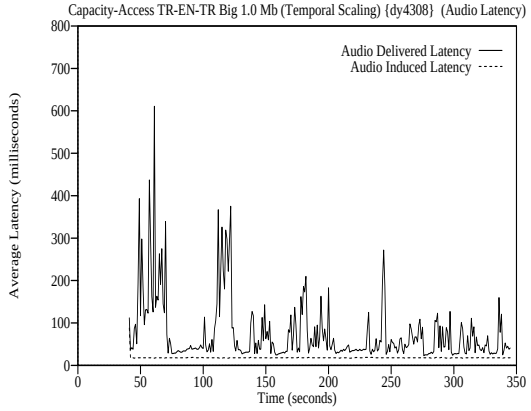
Figure 5-40: Combination Constraint (TR-EN-TR) Experiment 1 - Moderately Constrained - Video Scaling Only



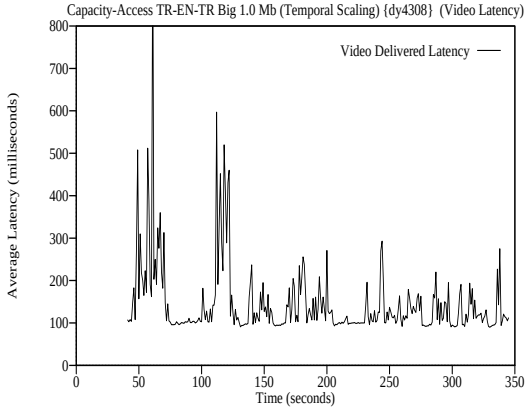
(a) Frames Per Second (FPS)



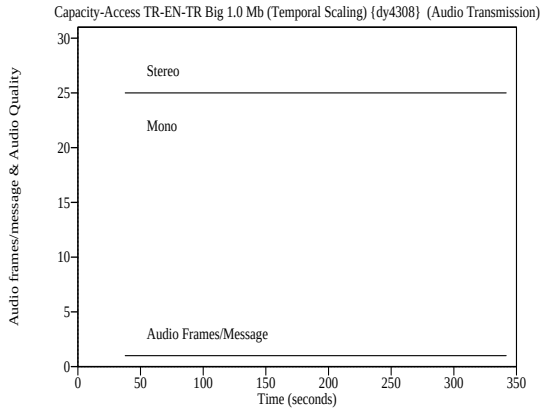
(b) Message Loss



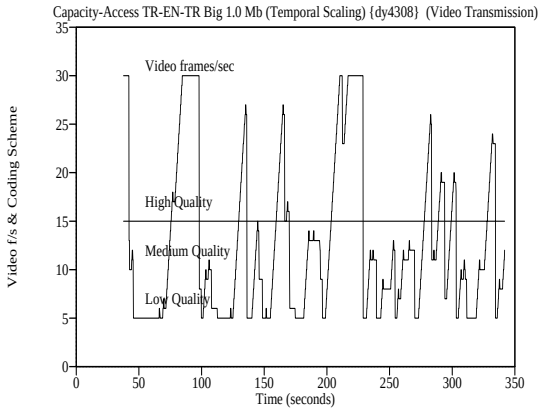
(c) Audio Latency



(d) Video Latency

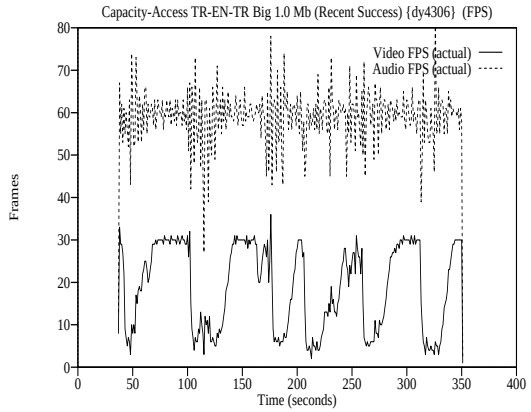


(e) Audio Transmission

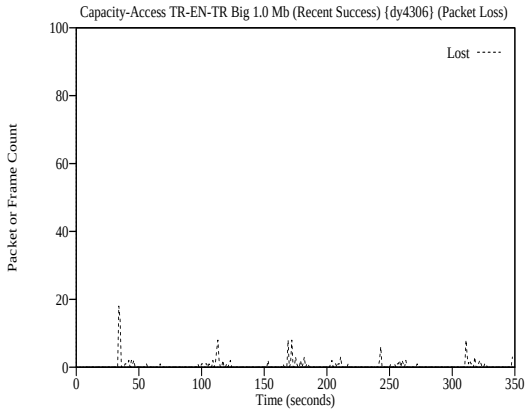


(f) Video Transmission

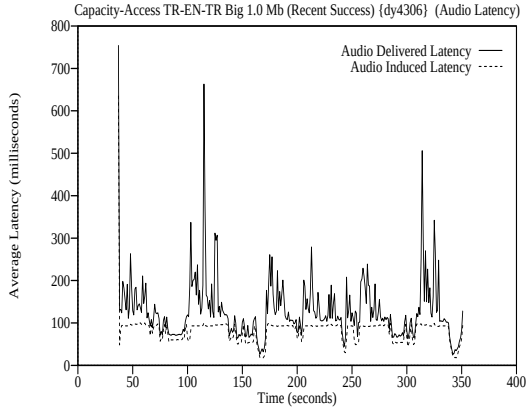
Figure 5-41: Combination Constraint (TR-EN-TR) Experiment 1 - Moderately Constrained - Temporal Scaling Only



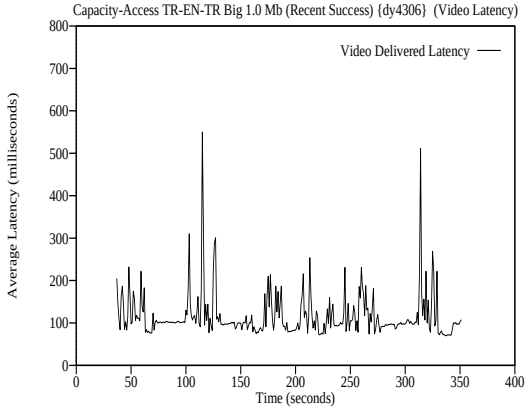
(a) Frames Per Second (FPS)



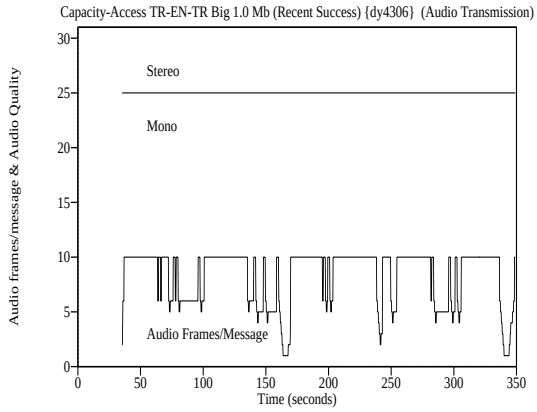
(b) Message Loss



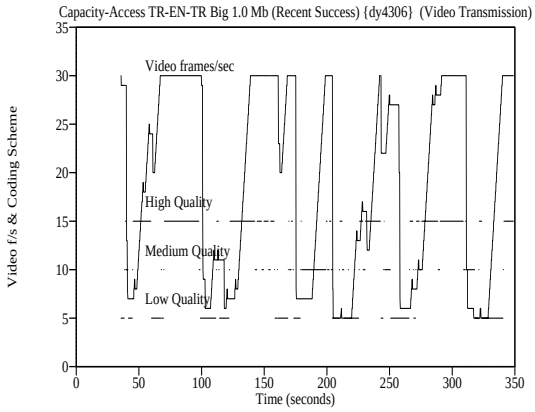
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-42: Combination Constraint (TR-EN-TR) Experiment 1 - Moderately Constrained - Recent Success

Capacity-Access	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	49.61	55.36	57.88	58.94
Standard deviation	14.44	9.70	6.89	6.99
Minimum	4	9	19	2
Maximum	63	70	75	86
Mode/Median	60/58	60/59	60/60	59/60
Gaps	3125	1383	625	332
Audio Latency:				
Delivered (Induced)				
Mean	121.87 (18.37)	93.80 (18.37)	76.25 (17.98)	127.14 (80.50)
Standard deviation	136.22 (0.95)	104.85 (0.74)	77.17 (0.43)	68.08 (21.01)
Minimum	32 (16)	23 (16)	23 (17)	26 (18)
Maximum	862 (22)	781 (21)	611 (19)	663 (100)
Mode/Median	39/50 (18/18)	34/60 (18/18)	28/44 (18/18)	71/113 (93/93)
Intervals > 250 ms	34	24	16	16
Audio Messages				
Frames Sent	18232	18431	18269	18866
Msgs(frames) Lost	3104 (3104)	1359 (1359)	597 (597)	52 (291)
Mean Frames Lost	10.18	4.41	1.96	0.92
Max Frames Lost	55	49	38	30
Video FPS				
Mean	23.87	27.39	12.15	18.77
Standard deviation	8.23	5.06	8.12	10.12
Minimum	2	3	1	1
Maximum	31	35	31	36
Mode/Median	30/29	30/30	5/10	30/20
Gaps	1846	778	5415	3513
Video Latency:				
Delivered				
Mean	184.25	148.52	146.15	116.61
Standard deviation	136.93	102.31	92.25	53.83
Minimum	97	74	90	70
Maximum	902	958	1024	550
Mode/Median	102/116	100/112	100/112	101/101
Intervals > 250 ms	44	27	24	8
Video Messages				
Frames Sent	9117	9215	3909	6044
Frames Lost	1835	766	178	114
Mean Frames Lost	6.02	2.49	0.58	0.36
Max Frames Lost	30	25	10	8

Table 5-14: Capacity-Access Constraint TR-EN-TR Experiment #1 Summary

5.6.2.3. Ethernet Combination Constraint - Experiment 2 Results

Figure 5-43, 5-44, 5-45, and 5-46 shows the results of experiment 2 for the BL, VSO, TSO, and RS transmission control schemes, respectively. Table 5-15 summarizes these results. Like experiment 1, experiment 2 has a varying capacity constraint at router *R1* (see Figure 5-14) and has a varying access constraint on the Ethernet. In this case, the access constraint is more severe than in experiment 1 and is generated by the Access Load 3 configuration (see Table 5-7).

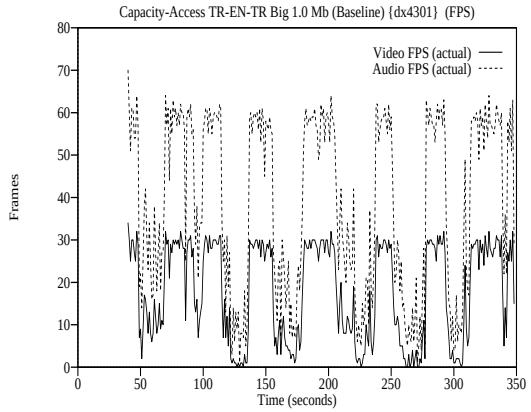
Figure 5-43 shows that the addition of another traffic generator on the Ethernet has a very negative impact on the performance of the BL algorithm. The conference experiences more frequent and more severe drops in frame delivery rates (part (a)). During the worst of these periods (*e.g.*, at approximately 260 seconds into the conference) audio is unintelligible for many consecutive seconds. The delivered video frame rate (and displayed frame rate) is essentially 0. Stream latencies during these periods are very high (over 700 *ms*; parts (c) and (d)) and many messages are lost (part (b)). Audio throughput is consistently poor and there are 6,558 audio gaps during the conference (Table 5-15). The average video frame rate is 18 FPS, but there are periods when no video is displayed (part (a) and Table 5-15).

VSO (Figure 5-44) does a much better job at delivering the conference than the BL algorithm, but the number of audio gaps almost doubles over the last experiment (2,510 gaps versus 1,383; Table 5-15). Audio and video latency both increase over that in experiment 1. Figure 5-44 (f) shows the VSO is sending almost exclusively low quality video.

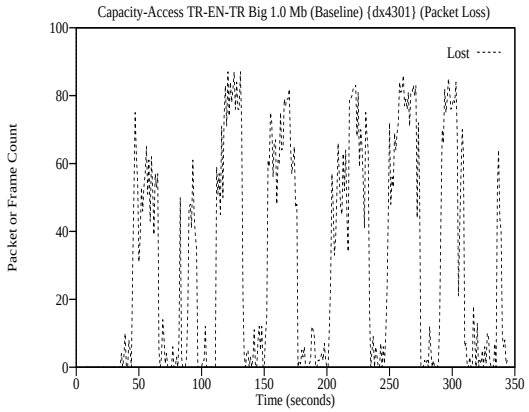
Performance also deteriorates with the TSO algorithm (Figure 5-45). The frame delivery rate drops for both streams (part (a)), message loss increases (part (b)), and stream latencies increase (parts (c) and (d)). The conference using TSO experiences 1,723 audio gaps. The average delivered video frame is only 9 FPS. Part (f) shows that the TSO algorithm is running out of adaptations to deal with the congestion; video frame generation tends toward the minimum rate of 5 frames per second.

Performance also degrades with the RS algorithm (Figure 5-46), but the degradation is more graceful than with the other algorithms. RS still significantly outperforms the other algorithms. The average delivered audio frame rate is 57.78 FPS and there are only 743 audio gaps (Table 5-15). Furthermore, under the additional load, RS also

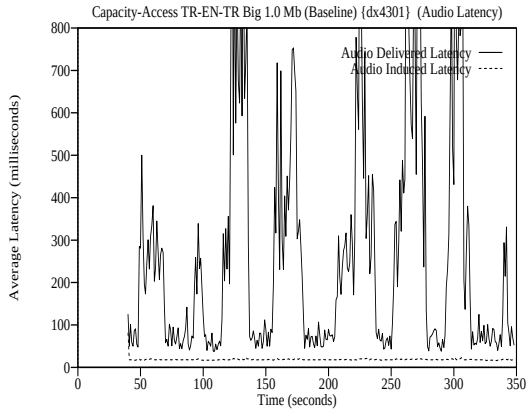
delivers audio with the lowest latency, even with the high induced latency resulting from audio packaging. The average video frame rate is still inferior to VSO (14 FPS versus 26 FPS), but is better than with TSO. The audio and video streams do not experience the intervals of extreme latency seen with the other algorithms (parts (c) and (d)).



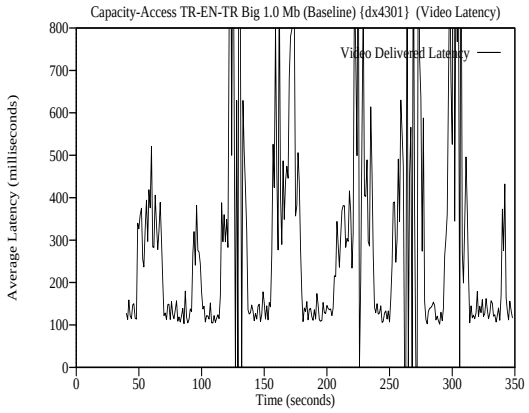
(a) Frames Per Second (FPS)



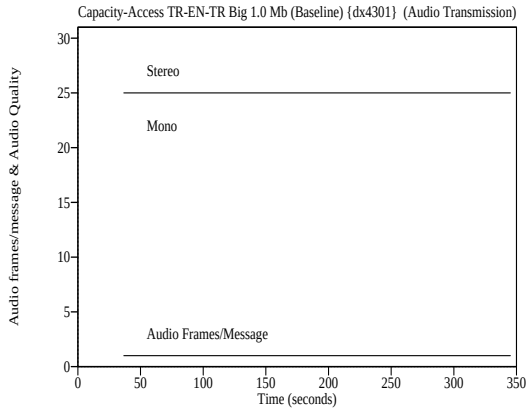
(b) Message Loss



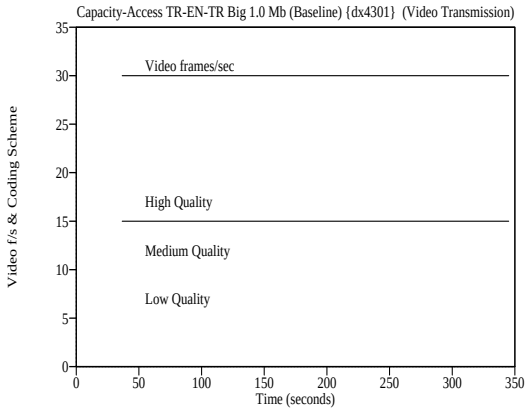
(c) Audio Latency



(d) Video Latency

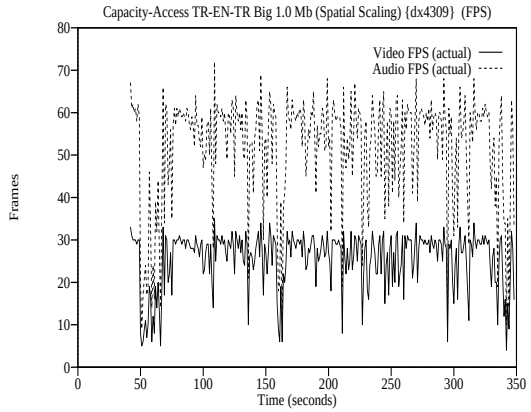


(e) Audio Transmission

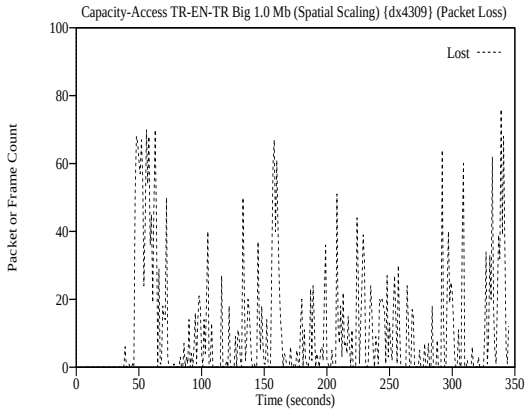


(f) Video Transmission

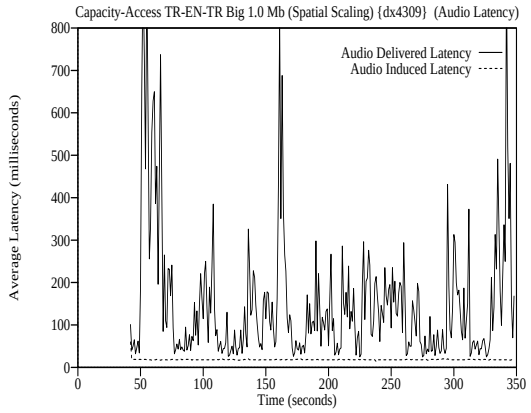
Figure 5-43: Combination Constraint (TR-EN-TR) Experiment 2 - Highly Constrained - Baseline



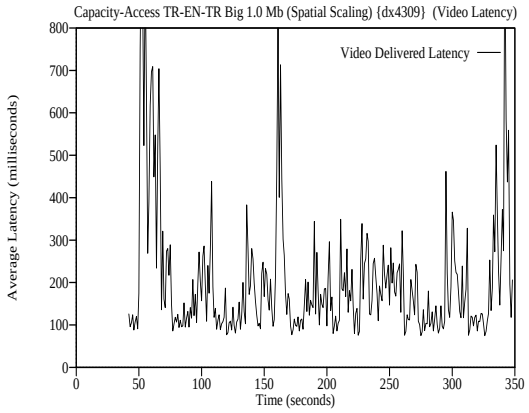
(a) Frames Per Second (FPS)



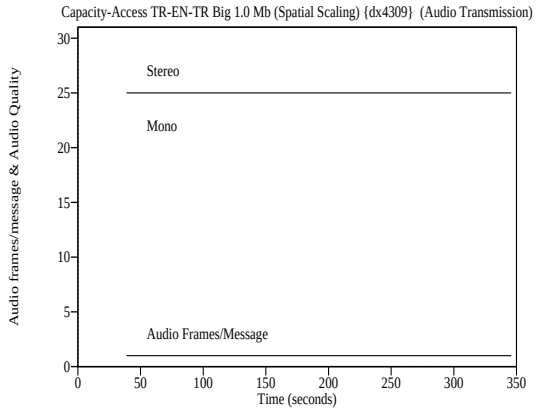
(b) Message Loss



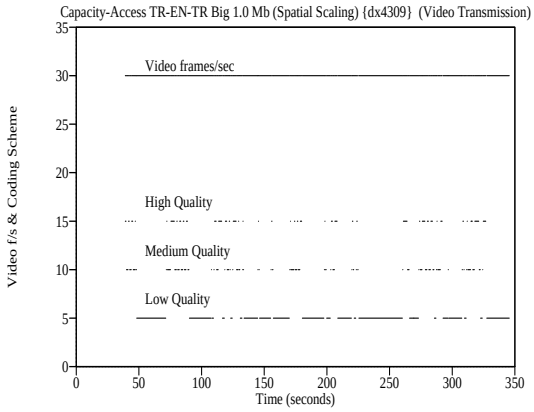
(c) Audio Latency



(d) Video Latency

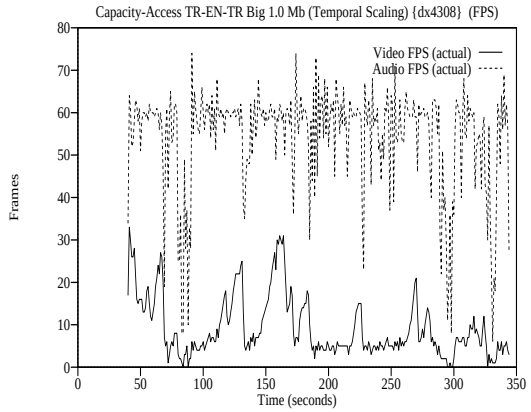


(e) Audio Transmission

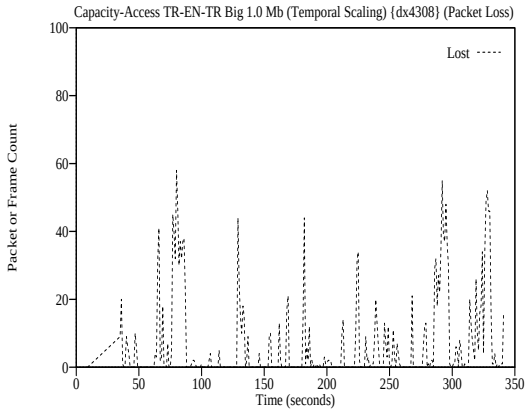


(f) Video Transmission

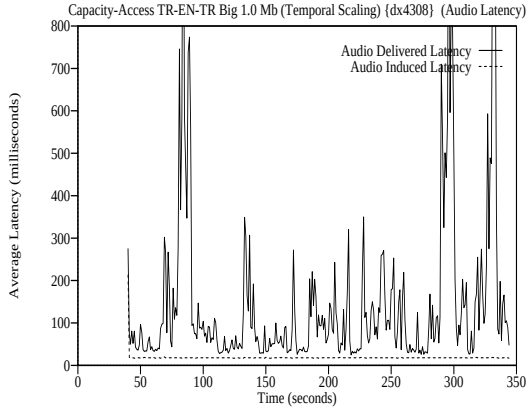
Figure 5-44: Combination Constraint (TR-EN-TR) Experiment 2 - Highly Constrained - Video Scaling Only



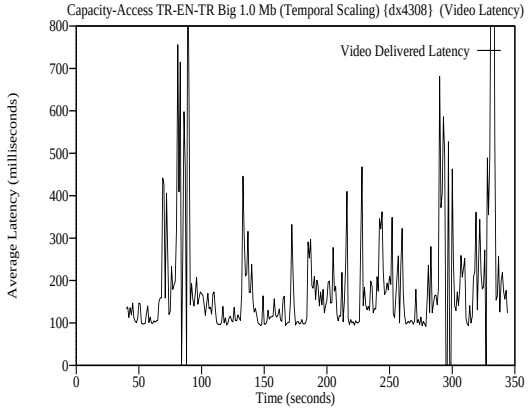
(a) Frames Per Second (FPS)



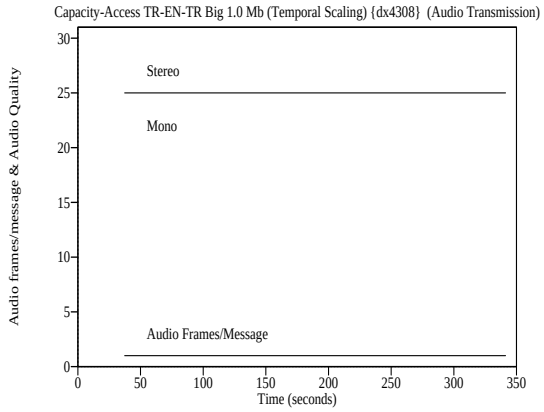
(b) Message Loss



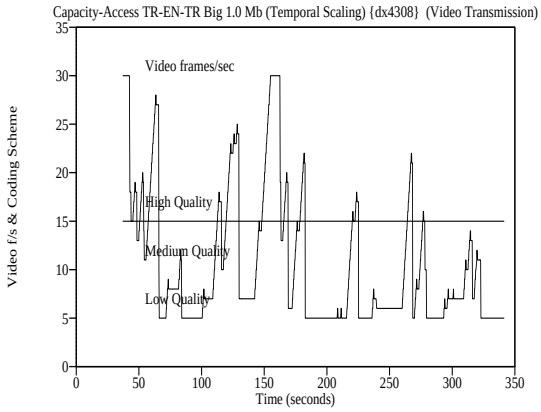
(c) Audio Latency



(d) Video Latency

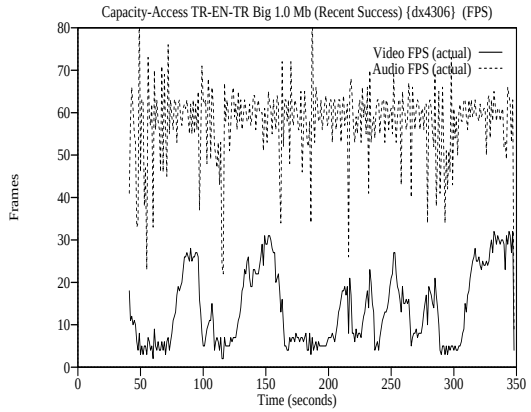


(e) Audio Transmission

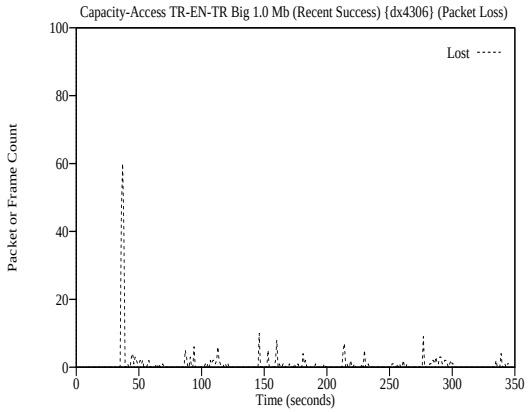


(f) Video Transmission

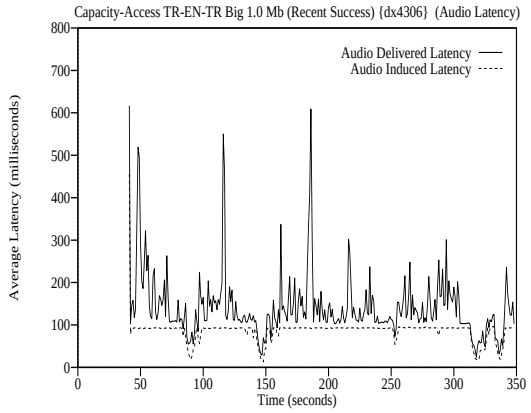
Figure 5-45: Combination Constraint (TR-EN-TR) Experiment 2 - Highly Constrained - Temporal Scaling Only



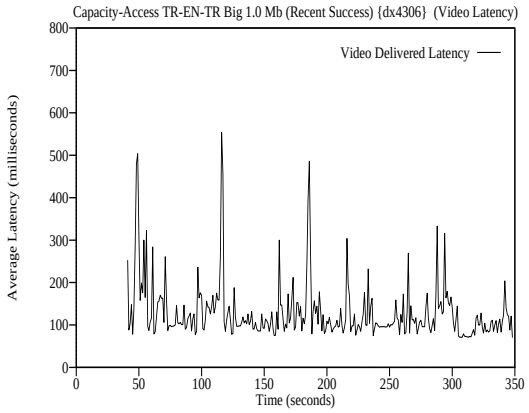
(a) Frames Per Second (FPS)



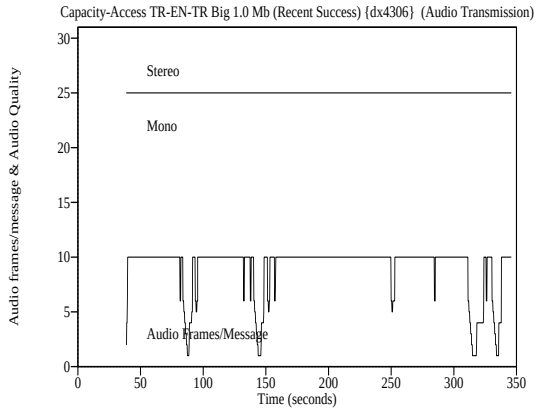
(b) Message Loss



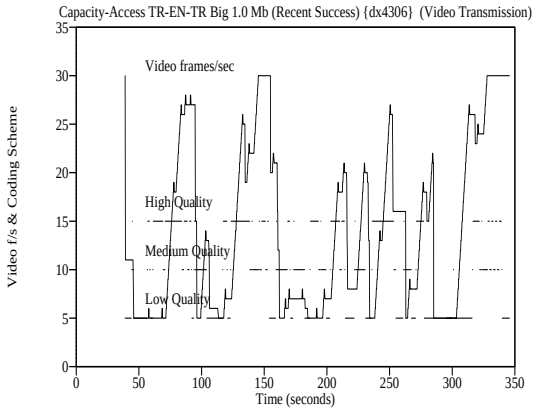
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-46: Combination Constraint (TR-EN-TR) Experiment 2 - Highly Constrained - Recent Success

Capacity-Access	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	38.63	51.77	54.34	57.78
Standard deviation	20.60	12.87	12.18	9.02
Minimum	1	9	6	9
Maximum	64	72	74	88
Mode/Median	59/42	60/57	60/59	63/59
Gaps	6558	2510	1723	743
Audio Latency: Delivered (Induced)				
Mean	264.43 (18.45)	152.90 (18.28)	147.69 (17.94)	140.47 (84.83)
Standard deviation	277.07 (1.16)	157.42 (0.65)	207.65 (0.43)	74.15 (19.03)
Minimum	38 (16)	25 (14)	25 (17)	24 (13)
Maximum	1343 (24)	944 (20)	1612 (20)	609 (96)
Mode/Median	50/137 (18/18)	50/105 (18/18)	29/81 (18/18)	107/122 (93/93)
Intervals > 250 ms	116	47	47	20
Audio Messages				
Frames Sent	18527	18434	18300	18550
Msgs(frames) Lost	6535 (6535)	2480 (2480)	1701 (1701)	163 (704)
Mean Frames Lost	21.08	8.05	5.56	2.27
Max Frames Lost	57	50	50	40
Video FPS				
Mean	17.67	25.57	9.24	13.69
Standard deviation	11.53	6.67	7.09	8.73
Minimum	0	4	0	2
Maximum	32	35	33	32
Mode/Median	30/20	30/28	5/6	5/11
Gaps	3789	1355	6313	5046
Video Latency: Delivered				
Mean	290.56	204.12	188.86	128.87
Standard deviation	261.87	156.62	154.78	69.20
Minimum	0	75	0	71
Maximum	1694	1035	1358	554
Mode/Median	0/160	118/152	100/141	96/107
Intervals > 250 ms	127	69	53	19
Video Messages				
Frames Sent	9264	9217	3216	4417
Frames Lost	3776	1339	369	170
Mean Frames Lost	12.18	4.35	1.21	0.55
Max Frames Lost	30	26	16	23

Table 5-15: Capacity-Access Constraint TR-EN-TR Experiment #2 Summary

5.6.2.4. Ethernet Combination Constraint - Experiment 3 Results

Figure 5-47, 5-48, 5-49, and 5-50 shows the results of experiment 3 for the BL, VSO, TSO, and RS transmission control schemes, respectively. Table 5-16 summarizes these results. Experiment 3 has combination constraints similar to those in experiments 1 and 2, but in experiment 3 one of the more passive traffic generators becomes more active (Access Load 4 in Table 5-7).

Figure 5-47 (a) shows that in this environment the performance with the BL algorithm is extremely poor. BL never delivers the full transmitted audio or video frame rate. Message loss is high throughout the conference (part (b)) and there are very long periods when the stream latencies are well above 250 *ms* (parts (c) and (d)). Average audio delivery is only 32.74 FPS and there are 8,296 audio gaps (Table 5-16). The average audio latency is 266 *ms*. The average delivered video frame rate is 14 FPS which is acceptable, but the average video latency is 325 *ms*.

Unfortunately, VSO can no adequately longer deal with the network congestion. Figure 5-48 shows that even though video is usually transmitted at the lowest image quality (part (f)), frame delivery is poor, particularly for the audio stream (part (a)). Average audio delivery is only 46.79 FPS and there are 2,165 audio gaps during the conference. Video is delivered at an average rate of 23 FPS, but the average latency is 259 *ms*. Both streams have many 20-40 second intervals where latencies are well above 250 *ms* (parts (c) and (d)). Message loss is high throughout most of the conference (part (b)).

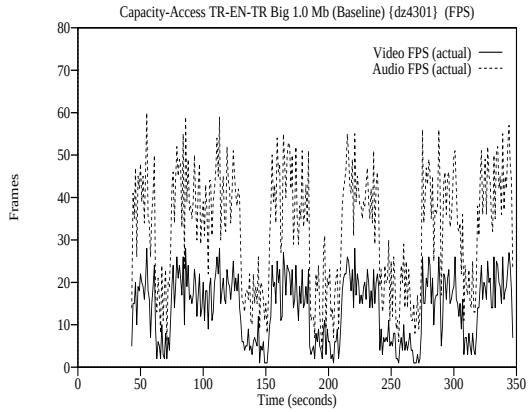
The TSO algorithm delivers a better audio stream (see Figure 5-49 (a)) than the BL and VSO algorithms. There are 930 audio gaps compared with the 2,165 with VSO (Table 5-16). The stream latencies are also improved, though there are still intervals with high latency (parts (c) and (d)). Furthermore, fewer messages are lost. TSO achieves these improvements by limiting the average video frame delivery rates to below 10 frames per second throughout most of the conference. The average delivered video frame rate is only 7 FPS and often borders on non-interactive.

Performance under RS (Figure 5-50) degrades gracefully in the face of the increased congestion. Audio stream quality remains good, with high frame delivery rates (part (b)) and acceptable latency (part (c)). Audio is delivered at an average rate of 58.20 frames per second with only 557 audio gaps. Audio and video latencies are better than

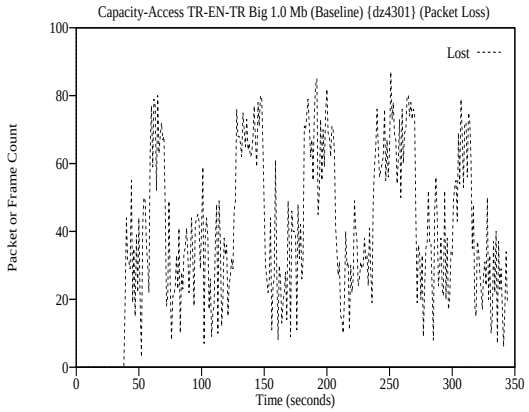
with any other algorithm. The average audio latency is 150 *ms* and the average video latency is 131 *ms*. Video quality varies with the level of congestion, but on average delivers about 14 FPS. RS only drops to very low video frame rates during the worst periods of congestion. There are a few intervals with high latency at the onset of periods with heavy congestion, but the intervals are short-lived and the average stream latencies are low. Even in the face of heavy congestion, RS keeps message loss very low (part (b)).

5.6.2.5. Conclusions for Combination Constraints on Ethernet Networks

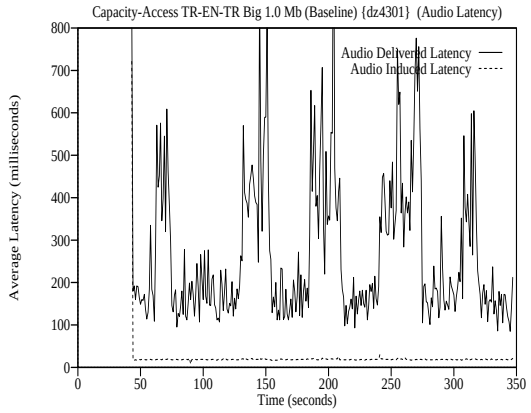
Under combination constraints, BL generally produces poor results and the performance deteriorates with increasing congestion. The scaling algorithms, VSO and TSO, deliver better conferences than BL. With low levels of congestion, the scaling algorithms can deliver conferences with competitive quality to those delivered by RS, but RS produces the best overall results even at low levels of congestion. As traffic load increases on the Ethernet, the differences in performance between the RS algorithm and the scaling algorithms increases. RS provides more graceful degradation under heavy congestion and delivers significantly better conferences under heavy congestion than either VSO or TSO. The RS algorithm is able to adapt to a wider range and degree of constraints because it has more options available for adaptation, including both bit and message rate changes. Both the bit and message rate changes are effective regardless of the number of hops separating the sender from the constraint or whether messages are fragmented.



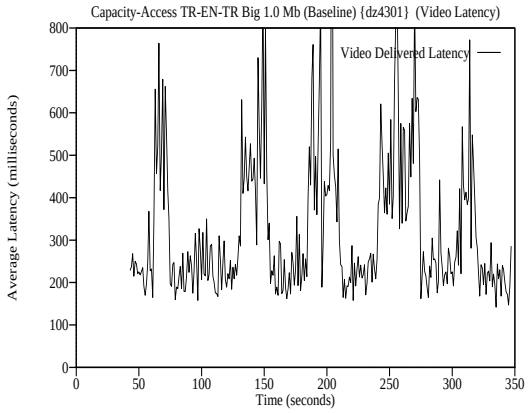
(a) Frames Per Second (FPS)



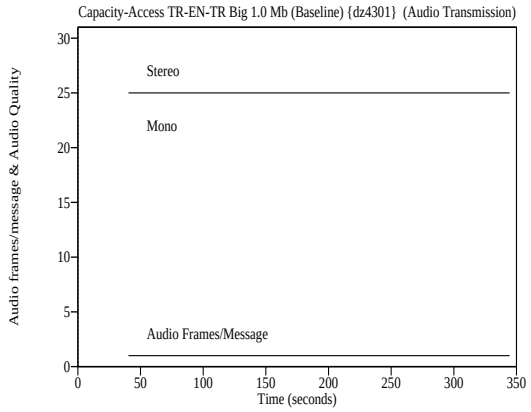
(b) Message Loss



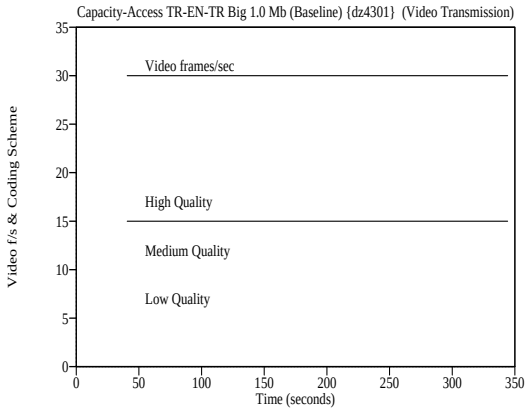
(c) Audio Latency



(d) Video Latency

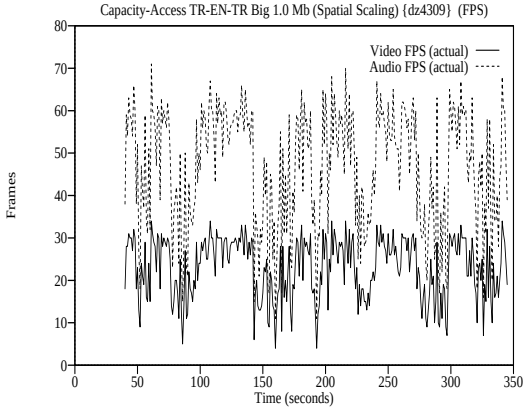


(e) Audio Transmission

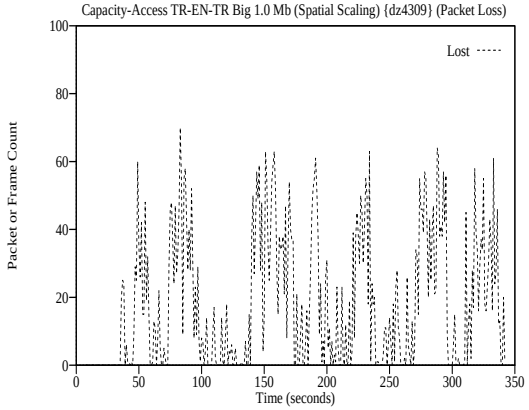


(f) Video Transmission

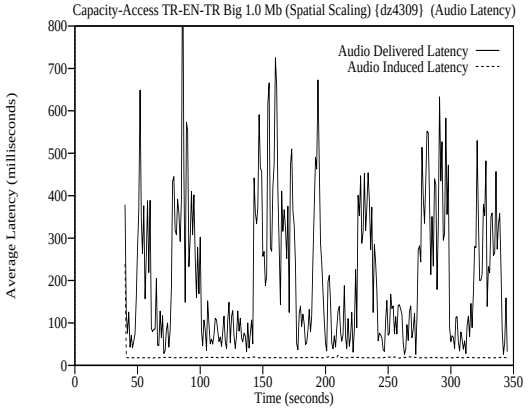
Figure 5-47: Combination Constraint (TR-EN-TR) Experiment 3 - Severely Constrained - Baseline



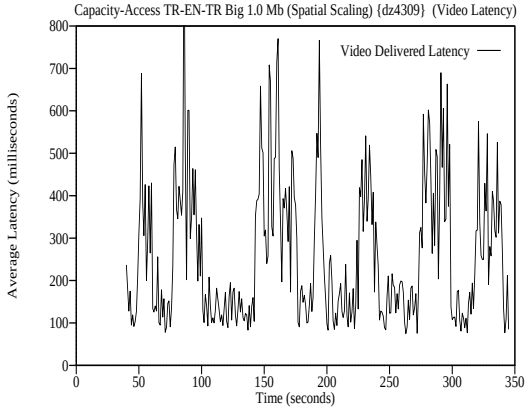
(a) Frames Per Second (FPS)



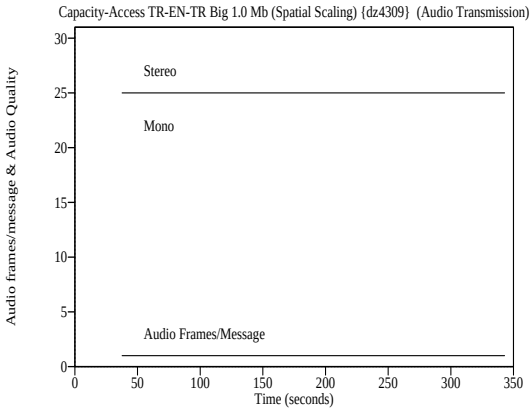
(b) Message Loss



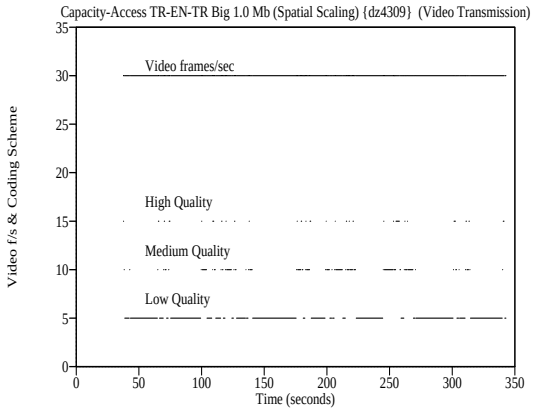
(c) Audio Latency



(d) Video Latency

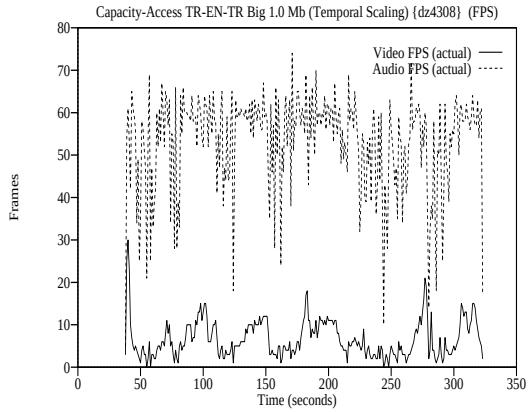


(e) Audio Transmission

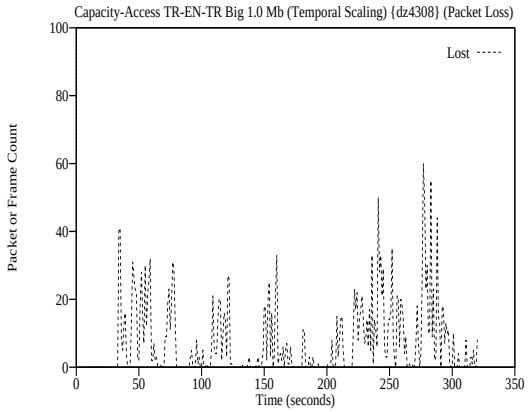


(f) Video Transmission

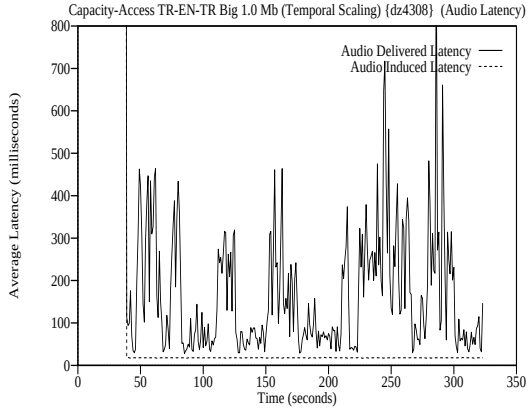
Figure 5-48: Combination Constraint (TR-EN-TR) Experiment 3 - Severely Constrained - Video Scaling Only



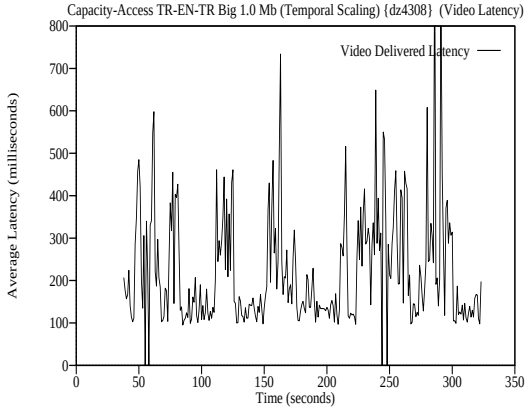
(a) Frames Per Second (FPS)



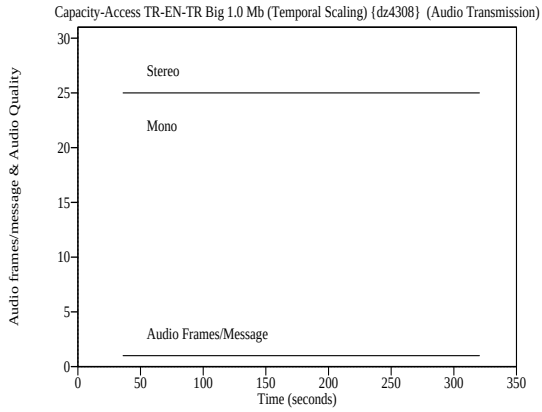
(b) Message Loss



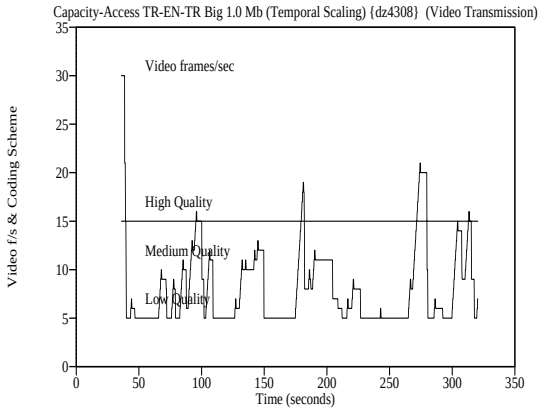
(c) Audio Latency



(d) Video Latency

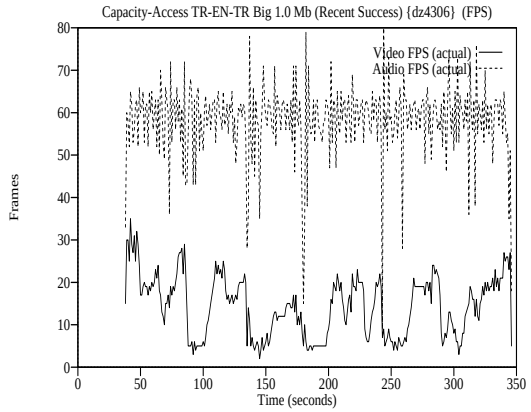


(e) Audio Transmission

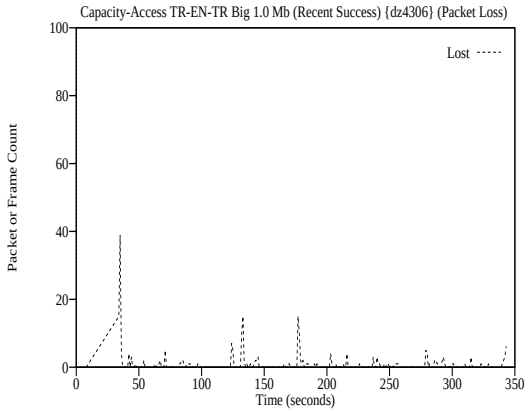


(f) Video Transmission

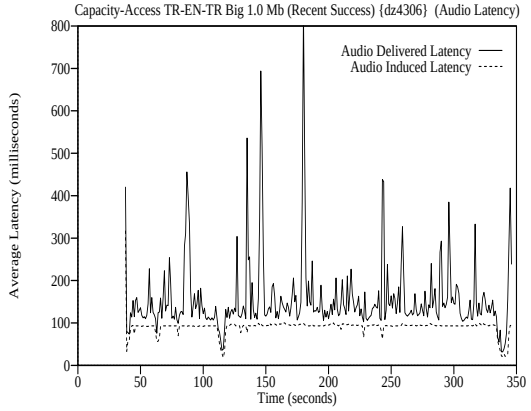
Figure 5-49: Combination Constraint (TR-EN-TR) Experiment 3 - Severely Constrained - Temporal Scaling Only



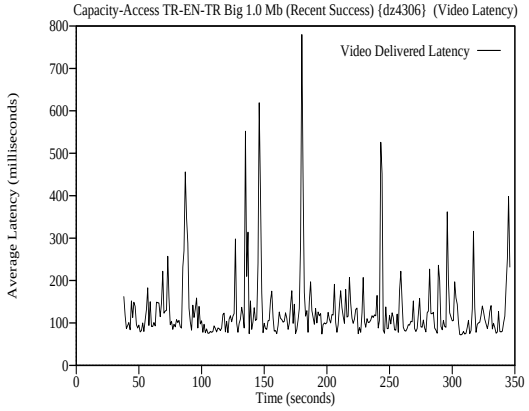
(a) Frames Per Second (FPS)



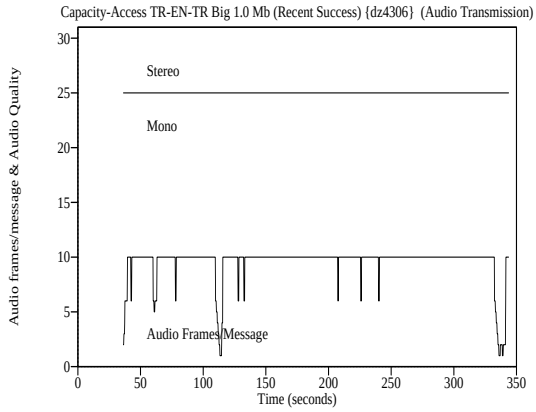
(b) Message Loss



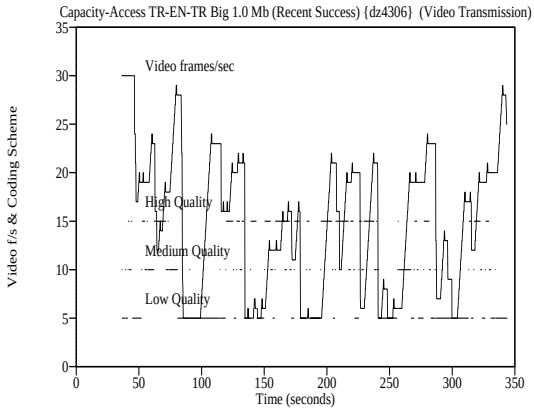
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 5-50: Combination Constraint (TR-EN-TR) Experiment 3 - Severely Constrained - Recent Success

Capacity-Access	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	32.74	46.79	53.32	58.20
Standard deviation	13.84	13.83	11.07	8.77
Minimum	4	11	10	9
Maximum	60	71	74	80
Mode/Median	42/35	60/50	60/57	63/60
Gaps	8296	2165	1930	557
Audio Latency: Delivered (Induced)				
Mean	266.41 (18.72)	211.78 (18.42)	166.32 (17.94)	155.26 (89.89)
Standard deviation	167.72 (1.42)	167.99 (0.67)	140.06 (0.36)	87.84 (14.66)
Minimum	85 (8)	26 (17)	28 (17)	32 (18)
Maximum	1139 (30)	1003 (24)	982 (19)	822 (101)
Mode/Median	112/191 (18/19)	76/143 (18/18)	38/113 (18/18)	110/132 (93/93)
Intervals > 250 ms	113	112	69	27
Audio Messages				
Frames Sent	18299	18398	17169	18493
Msgs(frames) Lost	8278 (8278)	4033 (4033)	1906 (1906)	91 (502)
Mean Frames Lost	27.05	13.14	6.64	1.62
Max Frames Lost	57	45	51	40
Video FPS				
Mean	13.92	22.96	6.67	14.36
Standard deviation	7.61	7.14	4.43	7.08
Minimum	1	4	0	2
Maximum	28	34	30	35
Mode/Median	15/15	30/24	5/5	5/15
Gaps	4896	4061	6660	4821
Video Latency: Delivered				
Mean	325.59	259.92	222.23	130.81
Standard deviation	171.08	166.32	138.41	84.33
Minimum	142	75	0	72
Maximum	1429	1076	1034	780
Mode/Median	189/254	91/195	111/176	87/106
Intervals > 250 ms	155	128	90	20
Video Messages				
Frames Sent	9149	9199	2340	4606
Frames Lost	4885	2149	408	155
Mean Frames Lost	15.96	7.00	1.42	0.50
Max Frames Lost	30	25	18	16

Table 5-16: Capacity-Access Constraint TR-EN-TR Experiment #3 Summary

5.7.3. Overall Combination Constraint Conclusions

The experiments with a combination of capacity and access constraints show that Recent Success (RS) consistently outperforms the Baseline (BL), Video Scaling Only (VSO), and Temporal Scaling Only (TSO) algorithms. RS is always superior from a network perspective. Under load, RS always has lower message loss rates than the competing algorithms. From a conference perspective, RS always delivers the best overall conference quality. The difference in performance between RS and the other algorithms is greater on networks with large MTUs and heavier loads, but RS is superior in all environments. RS always delivers equivalent or better audio fidelity to that delivered by the other algorithms. The VSO algorithm sometimes delivers higher video frame rates, but at the expense of the more important audio stream. TSO sometimes delivers equivalent audio fidelity to that of RS, but with inferior video stream quality. RS does a much better job than the other algorithms at controlling latency. Video latency is always equivalent or lower with RS than with the other algorithms. Absolute audio latency is sometimes higher with RS than with the other algorithms because of the induced latency caused by audio frame packaging, but RS does not suffer the long periods of excessive latency seen with the other algorithms. Audio latency with RS is almost always below our latency guideline. RS makes an explicit tradeoff between audio latency and frame delivery. The result of this tradeoff is that audio latency is controlled, but exploited to produce higher delivered frame rates and lower loss rates than with the other algorithms. Similarly, RS sometimes voluntarily reduces video frame rates and video image quality to improve the overall delivered quality of the conference and to limit message loss.

5.7. Overall Conclusions from Controlled Network Experiments

The sections of this chapter have discussed a set of experiments evaluating the performance of the Baseline, Video Scaling Only, Temporal Scaling Only, and Recent Success algorithms under access constrained networks, capacity constrained networks, and networks with a combination of access and capacity constraints. Table 5-17 gives a qualitative summary of the results of all the experiments in this chapter.

	Network Configuration		
Constraint Type	(A) Token Ring Backbone (MTU = 17,800 bytes)	(B) Token Ring Backbone (MTU = 1,500 bytes)	(C) Ethernet Backbone
(1) Access	BL, VSO, TSO, RS (HBR) **Poor performance with BL, VSO, and TSO.** **RS delivers significantly better results than other algorithms.**	BL, VSO, TSO, RS (HBR) **Very poor performance with BL, VSO, and TSO.** **RS results inferior to those in A1, but significantly better than other algorithms.**	(a) BL, VSO, RS (HBR) moderately constrained **Perceptual draw between VSO and RS. VSO has better video, but RS has better audio.** (b) BL, VSO, RS (LBR) moderately constrained **RS is superior to BL and VSO.** (c) BL, VSO, RS (LBR) highly constrained **RS is far superior to BL and VSO.**
(2) Capacity	BL, VSO, TSO, RS (HBR) **RS delivers the best results. VSO delivers slightly better video than RS, but inferior audio. TSO delivers competitive audio, but inferior video.**	BL, VSO, TSO, RS (HBR) **Results are perceptually similar to those in A2. RS delivers the best overall conference.**	Results similar to B2; results not shown

Table 5-17: Qualitative Summary of Chapter 5 Experiments

	Network Configuration		
Constraint Type	(A) Token Ring Backbone (MTU = 17,800 bytes)	(B) Token Ring Backbone (MTU = 1,500 bytes)	(C) Ethernet Backbone
(3) Both Access and Capacity	BL, VSO, TSO, RS (HBR) **Very poor performance from BL, VSO, and TSO.** **RS far outperforms the other algorithms.**	Not evaluated.	(a) BL, VSO, TSO, RS (HBR) moderately constrained **VSO has better video frame rates, but RS has better audio. TSO has competitive audio, but inferior video.** (b) BL, VSO, TSO, RS (HBR) highly constrained **RS has much better audio than others and better control of latency. VSO has better video frame rates than RS.** (c) BL, VSO, TSO, RS (HBR) severely constrained **RS has much better audio than others and better latency control. Difference between VSO video and RS video less than in (a) and (b).**

Table 5-17 (continued): Qualitative Summary of Chapter 5 Experiments

The net result of all these experiments is that the Recent Success algorithm outperforms the other candidate algorithms in all environments. This fact is not surprising since Recent Success is capable of video scaling as well as packaging. The scaling algorithms do not exploit packaging. This combination of scaling and packaging techniques makes Recent Success successful in all network environments with all types of constraints.

The addition of a packaging strategy makes Recent Success particularly effective on access constrained networks when compared with the scaling algorithms. We believe dynamic access constraints are far more common on existing LANs than are dynamic capacity constraints. Access constraints are easier to create on existing local area networks than capacity constraints. We have also shown that even when capacity constraints are present in the network, their presence may lead to secondary access constraints. Capacity constraints, on the other hand, are not generated as a result of access constraints. Typically, capacity constraints are caused by physical limits in capacity (*i.e.,* are structural capacity constraints) rather than dynamic capacity constraints caused by variance in the bit processing rate in the network components. The implication of this is that on existing local area networks it is often more important to address access constraints than capacity constraints. Transmission control algorithms that employ only video scaling do not directly address access constraints. Scaling either addresses access constraints by reducing the bit rate of the video stream (*e.g.,* with spatial scaling), which only indirectly addresses the packet rate produced by the stream, or by reducing the video frame rate (*e.g.,* with temporal scaling), which may unnecessarily reduce the delivered video fidelity. On the other hand, the transmission control framework and the associated RS algorithm are much better suited to dealing with access constraints due to the manipulation of message rates without sacrificing the possibility of reducing the bit rate when faced with capacity constraints. RS may be able to address access constraints solely through packaging changes on the audio and video streams. For example, in the token ring access constraint experiments, we demonstrated an environment where almost all the effects of the access constraint could be ameliorated by adaptation of the audio stream alone. Note that on token rings, frame packaging could also be applied (though it was not in these experiments) to the video stream since the large MTU on token rings allow multiple video frames to be packaged in a single token ring packet.

The major conclusion here is that algorithms such as Recent Success that are based on the transmission control framework are better able to handle congestion than algorithms solely relying on stream scaling and the prevalence of access constraints on LANs emphasizes the advantages of adding a packaging strategy to bit rate control. The degree to which a two-dimensional transmission control scheme outperforms other schemes increases with increasing network congestion. The presence of fragmentation may have no negative effect on the use of packaging depending on the relative location of fragmentation and the congestion constraint. Even if fragmentation

occurs well before the constraint (the worst case from a packaging perspective), packaging is still more effective than not performing packaging because decreasing message rates in response to the congestion constraint leads to an efficient packing of media frames into packets when fragmentation does occur. The experiments in this chapter show that in controlled network experiments, Recent Success always produces conferences with better overall fidelity, latency, and message delivery than other algorithms. The next chapter demonstrates that this is also the case on a production network.

Chapter VI
Production Network Experiments

6.1. Purpose of the Production Network Experiments

This chapter discusses a set of experiments run on a production network. A production network is a network used by a group of computer users to do a set of tasks. In these networks, we cannot control or predict the amount of traffic on the network. Examples of production networks include networks used to support businesses, universities, or other organizations. Production networks vary in topology, traffic patterns, number of users, network technologies, *etc*. In this chapter, we only use production networks that do not support reservation of network resources. Most current networks do not provide for resource reservation.

It is difficult to model production networks using traditional queueing networks. There is no widely accepted traffic model for production networks and some research has implied that the mathematical models most often used to represent traffic patterns are inadequate and inaccurate [71]. Identifying and validating a model of production network traffic is beyond the scope of this dissertation. If such a model existed, we could evaluate the transmission control framework directly against the accepted traffic model. Unfortunately, no such model exists. The approach taken in this dissertation is to first describe why the transmission control framework is an effective strategy for dealing with network congestion (Chapter 3), then demonstrate the effectiveness of the framework using controlled network experiments (Chapter 5). Finally, the experiments in this chapter demonstrate that the transmission control framework can effectively deal with congestion on a production network. Together, Chapters 5 and 6 show that algorithms based on the framework outperform less sophisticated algorithms in both controlled network experiments and production network experiments.

This chapter is organized as follows. First, we describe the production network used for the experiments in this chapter. Next, we describe a series of experiments performed over several days and using the Baseline (BL), Video Scaling Only (VSO), Temporal Scaling Only (TSO), and Recent Success (RS) transmission control

algorithms. Table 6-1 shows the dates of the experiments and the control algorithms used for the experiments. An "x" in the table indicates that we used the control strategy in the experiments performed on a particular date.[1] We compare the results of conferences using different control algorithms under a variety of load conditions. Finally, we discuss our conclusions.

Date of Experiment	BL	VSO	TSO	RS
April 18, 1995	x	x		x
April 19, 1995	x	x		x
April 20, 1995 (set 1)	x	x		x
April 20, 1995 (set 2)	x	x		x
April 21, 1995	x	x		x
April 24, 1995	x	x		x
April 25, 1995	x	x		x
April 26, 1995	x	x	x	x
April 28, 1995	x	x	x	x

Table 6-1: Overview of Production Network Experiments

6.2. Experiment Environment and Procedure

Figure 6-1 shows the production network used for the experiments in this chapter. The source and destination conferencing machines are the same as those used in the previous chapters. The source and destination machines are connected to unloaded token ring networks (*i.e.*, the only traffic on the token rings is that generated by the video conference) and are connected to the production network via two routers, *R1* and *R2*, that are IBM RS/6000 workstations running AIX 3.25. The production network consists of multiple Ethernet segments connected by a Wellfleet bridge/router. All the computers in the Department of Computer Science at the University of North Carolina at Chapel Hill (UNC) are connected to the Ethernet segments and all the components of the production network are contained in a single

[1]Ideally, we would have used the BL, VSO, TSO, and RS algorithms in every experiment. Unfortunately, we only decided to include TSO after we had run already run several experiments, so TSO is only included in the experiments on April 26 and April 28.

building, Sitterson Hall, on the UNC campus. For convenience, we refer to the production network as the *Sitterson network*. The production traffic on the Sitterson network consists of the traffic generated by the normal activities of a computer science department at a research university (*e.g.*, conducting research projects, performing department administrative functions, doing class work, reading network news, sending and receiving electronic mail, *etc*.). The traffic load varies widely and unpredictably. Each experiment sends an audio and video conference from the source machine to the destination machine via the routers and the Wellfleet (the conference traffic is represented by the thick arrow in Figure 6-1). The routers *R1* and *R2* are connected to different Ethernet segments, so all traffic must pass through the Wellfleet. Feedback from the conference destination to the conference source follows the reverse path back to the conference source (feedback is represented by the thin arrow in Figure 6-1). We ran five minute conferences using several transmission control algorithms, measured the results, and summarized the results in a set of graphs.

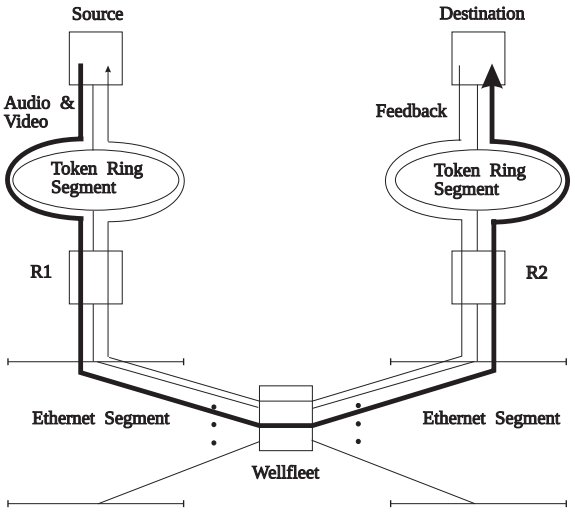


Figure 6-1: Sitterson Network

6.3. Expected results

Given the results from earlier chapters of this dissertation and some characteristics of our production network, we can make some general predictions about the expected results from the production network experiments. First, since traffic is uncontrolled

on the production network, experiments run at different times may experience dramatically different levels of network congestion. Even when experiments are separated by only minutes, the levels of congestion may differ significantly. We must carefully examine the results to determine the degree to which we can compare different experiments. In this chapter, we only compare experiments run within minutes of each and even then we are careful to assess the degree to which the traffic loads are comparable.[2]

The lack of control on the production network traffic means it is generally not possible to exactly reproduce an experiment on the production network. The lack of reproducibility is the major drawback to experimenting with a production network. Unfortunately, techniques such as recording the network traffic and simulating different strategies with the resulting trace [103] are not possible when testing transmission control schemes since the decisions made by the transmission control policy may affect the traffic pattern. The strategy taken here is to run the experiments across a number of days and compare the overall results.

The Sitterson network is managed by a staff of network administrators. Since the network is a critical resource within the computer science department, the network administrators spend a lot of time making sure the network performs well. Unfortunately, this section demonstrates that even well run production networks may experience loads that severely impact video conferencing.

Figure 6-2 shows the operating points for the video conferencing system used in the experiments in this chapter. This is the same conferencing system as the High Bit Rate (HBR) system used in the experiments in Chapter 5 except that for perceptual reasons we eliminated all audio and video operating points with message (or frame) rates below 5 frames per second.

---

[2]Ideally, we would have directly measured the traffic loads at the Wellfleet and on the Ethernet networks. Unfortunately, at the time these experiments were performed, we did not have the appropriate monitoring hardware and software to measure these loads. Based on conversations with the primary network administrator for the Sitterson network, we believe that the Sitterson Ethernets are often access constrained and that most packet loss in the network occurs at the Wellfleet router [115].

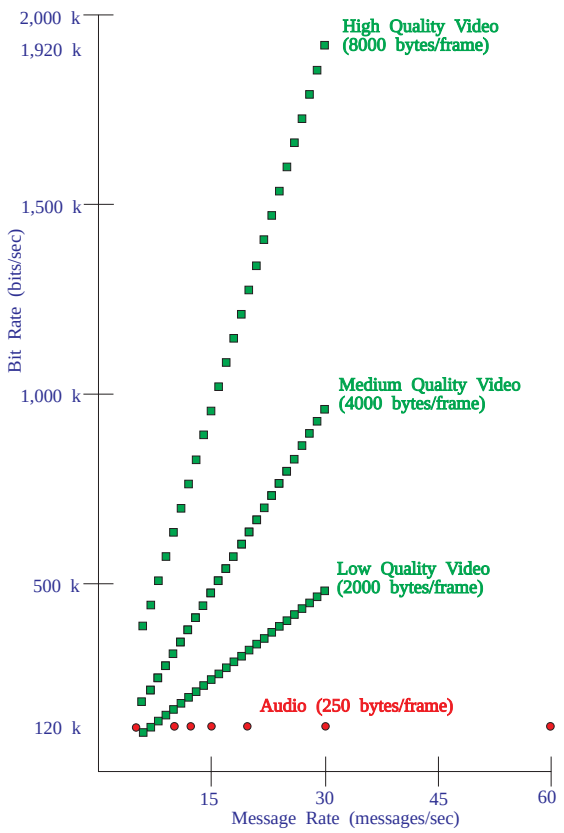


Figure 6-2: Operating Points for the Conferencing System used in Chapter 6

Since the Sitterson network is composed primarily of Ethernet segments, the video frames in our experimental system cannot be carried in a single network packet and only 6 audio frames may be packaged in a single network packet. The transmission control algorithms have no direct knowledge of these limits. Router *R1* fragments any messages greater than 1,500 bytes. Figure 6-3 shows the realization of the operating points in Figure 6-2 on the Ethernet segments of the network in Figure 6-1.

As discussed in Chapters 3 and 5, the fragmentation on Ethernet makes video the dominant packet generator (of the conference streams) on the Ethernet segments and somewhat limits the effect of packaging decisions by the Recent Success algorithm. Since the Ethernet segments are the only segments with production traffic (the token rings are unloaded), congestion can only occur on these segments or at the routers. In this environment, we expect the relative differences between the performance of algorithms based on the transmission control framework and those based on video scaling to be smaller than in an environment where there are more packaging options. Even if the primary constraint is an access constraint, we expect audio packaging to

have a smaller impact than changes to the video stream because of the relatively large contribution of packets by video compared with audio. Put another way, we expect there will be less difference in performance between conferences using the Recent Success algorithm and those using scaling algorithms on the Sitterson network than on networks composed entirely of token ring segments. RS will not be able to exploit packaging to the same degree on the Sitterson network as on an all token ring network because of the relatively small MTU on the Sitterson Ethernets.

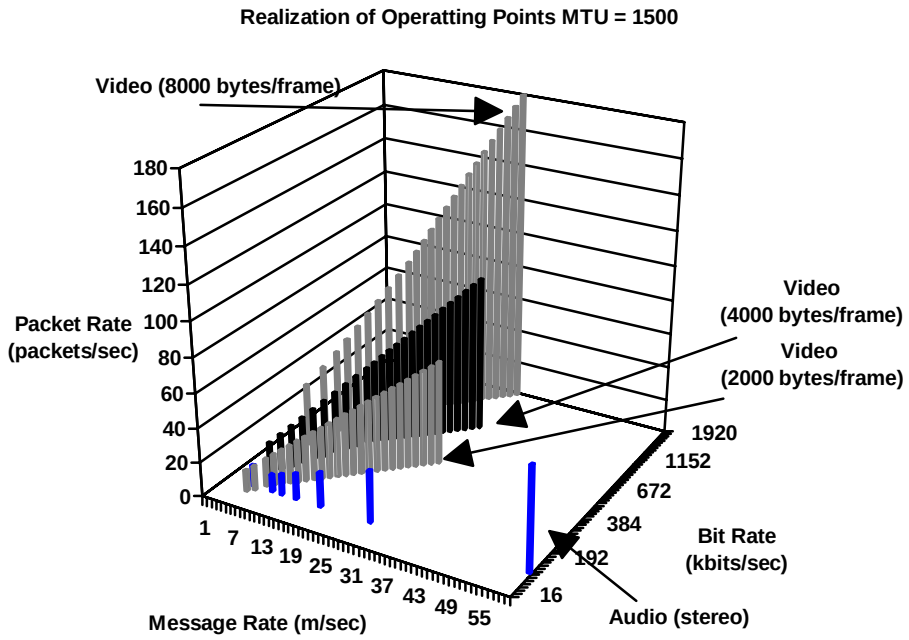


Figure 6-3: Realization of Figure 6-2 Operating Points on Ethernets in Figure 6-1

6.4. Experimental results

This section describes the results measured for several experiments on the Sitterson network. We ran the experiments between April 18, 1995 and April 28, 1995. All experiments were on normal work days. We discuss the test measurements and conclusions for each experiment, then we present an overall set of conclusions.

6.4.1. Results for Tuesday, April 18, 1995

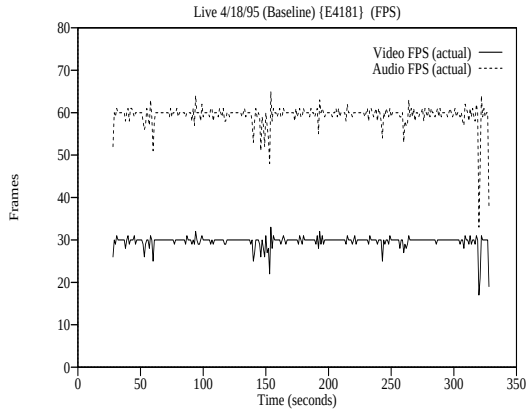
Figures 6-4, 6-5, and 6-6 show the results for conferences using the Baseline (BL), Video Scaling Only (VSO), and Recent Success (RS) algorithms, respectively. Table

6-2 gives the summary statistics for the three conferences. We ran the three conferences one after another between 11:00 to 11:30 a.m. on April 18, 1995. In this particular experiment, there was very little traffic on the production network other than the video conference traffic. We can infer this from the low latency and low loss experienced with the BL algorithm. Figure 6-4 shows that when the network is lightly loaded, a naive transmission strategy like BL works very well. The delivered audio and video frame rates (part (a)) are nearly perfect. Audio and video latencies are low (parts (c) and (d)). There is occasional message loss (part (b)), but only a relatively small number of messages are lost and the loss is relatively infrequent. The overall quality of the conference is nearly indistinguishable from an unloaded standalone network.

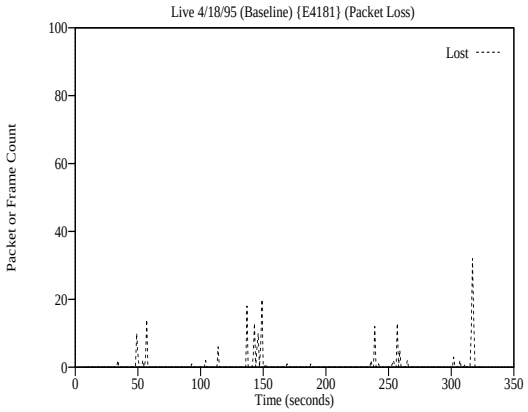
Figure 6-5 shows the results for the VSO algorithm. The performance is quite similar to that observed with the BL algorithm. Delivered frame rates are nearly perfect (part (a)) and latencies are low (parts (c) and (d)). Video quality is sometimes intentionally lowered during the course of the conference (part (f)), which results in slightly less message loss (part (b)) from that experienced with the BL algorithm.

Figure 6-6 shows the results for the RS algorithm. Like BL and VSO, RS delivers high frame rates (part (a)) and low latencies (parts (c) and (d)). Message loss is even lower than with BL and VSO (part (b)). Video is almost always transmitted using a high quality encoding (part (f)). Audio frames are usually packaged one frame per message, although occasionally more frames are packaged together (*e.g.*, between 220-250 seconds into the experiment on part (e)), but audio latency never exceeds approximately 100 milliseconds, even with the induced latency associated with this packaging (part (c)).

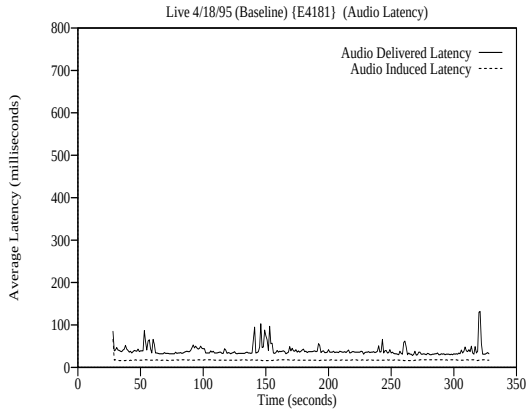
The first conclusion from this experiment is that on a lightly loaded LAN almost any strategy for transmitting audio and video will work. A second conclusion is that there is no performance penalty for using a well-designed adaptive transmission algorithm, such as VSO or RS, even when the network is lightly loaded and well behaved. Even with the lightly loaded network in this experiment, VSO and RS outperform BL, although the performance differences are minor. Overall, RS outperforms both BL and VSO in this experiment. Table 6-2 shows that the conference using RS has fewer gaps and less message loss than conferences using BL or VSO. In Chapter 5, we showed RS is superior to BL and VSO under heavy network loads and this experiment demonstrates that RS also performs well on lightly loaded networks.



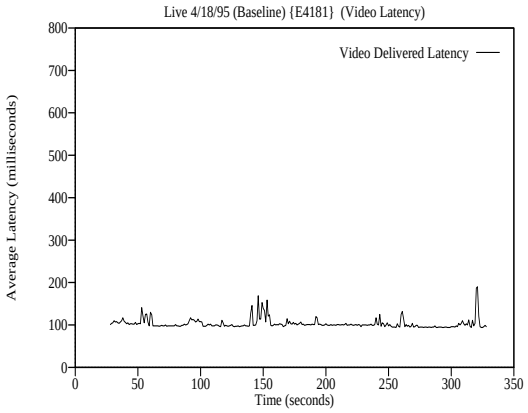
(a) Frames Per Second (FPS)



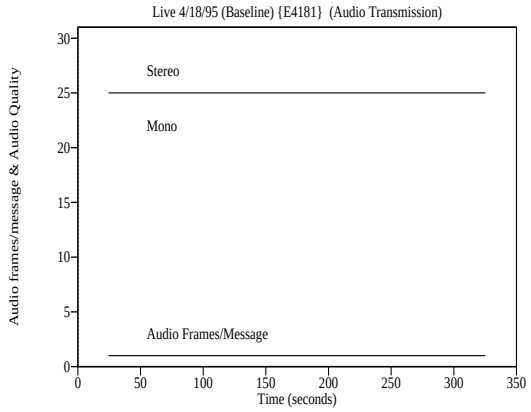
(b) Message Loss



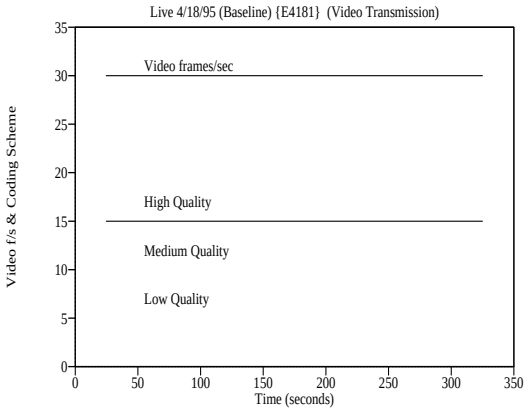
(c) Audio Latency



(d) Video Latency

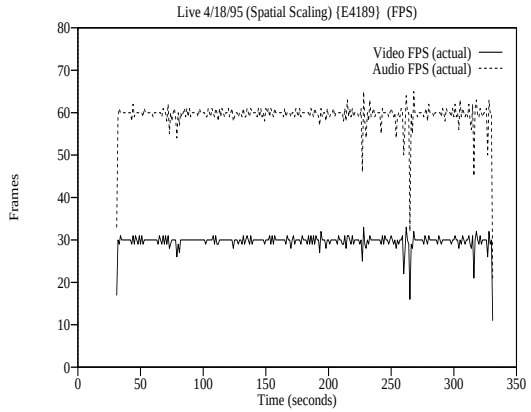


(e) Audio Transmission

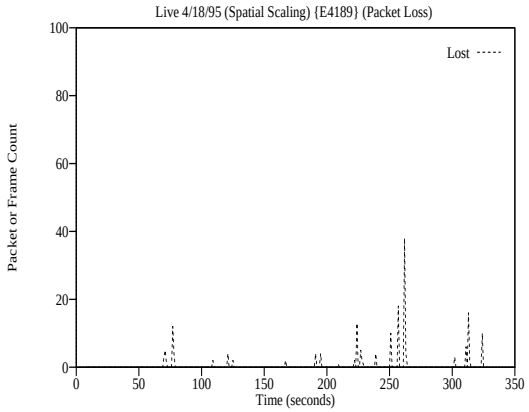


(f) Video Transmission

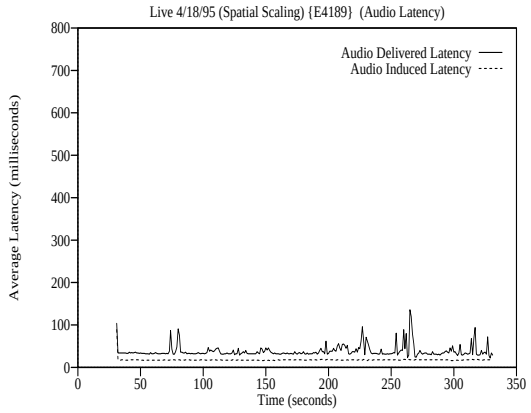
Figure 6-4: Sitterson Network Experiment - April 18, 1995 - Baseline



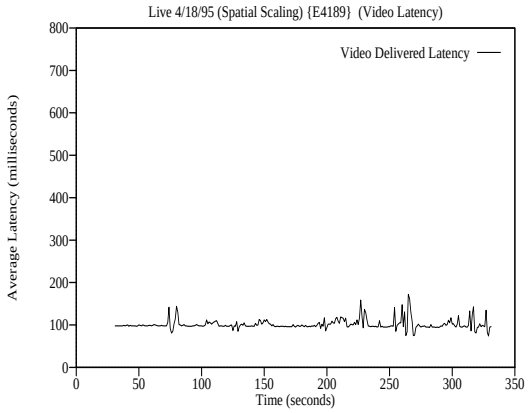
(a) Frames Per Second (FPS)



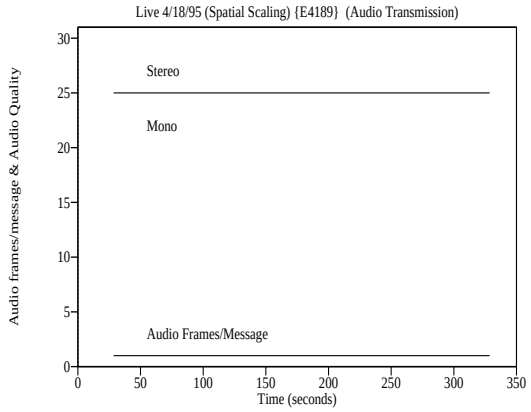
(b) Message Loss



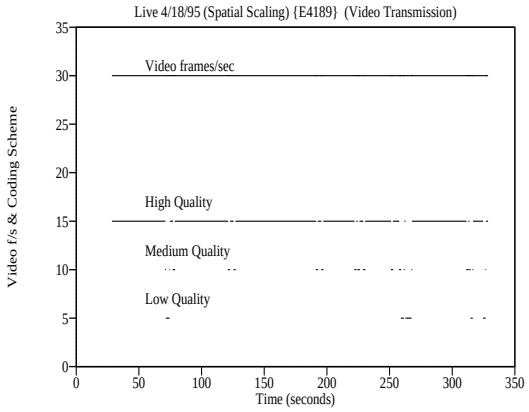
(c) Audio Latency



(d) Video Latency

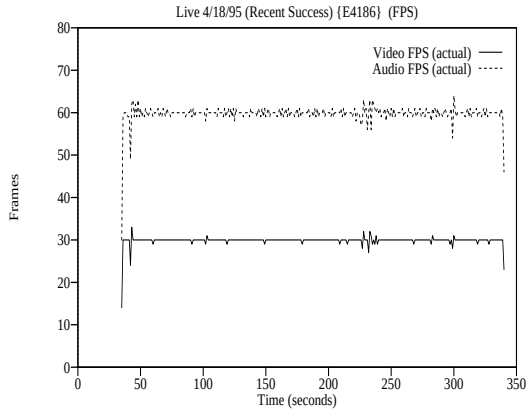


(e) Audio Transmission

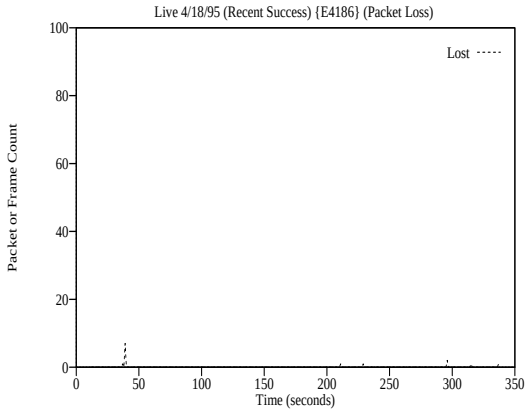


(f) Video Transmission

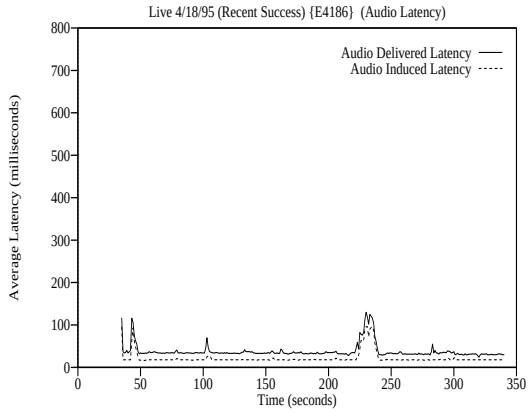
Figure 6-5: Sitterson Network Experiment - April 18, 1995- Video Scaling Only



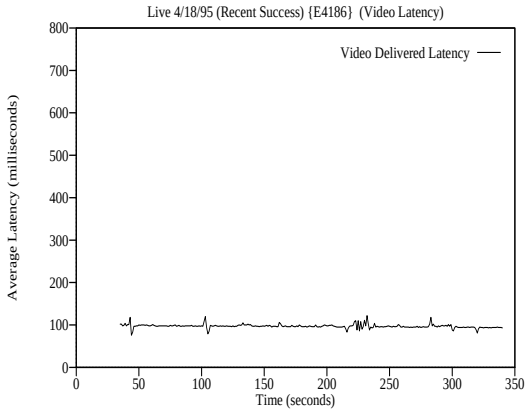
(a) Frames Per Second (FPS)



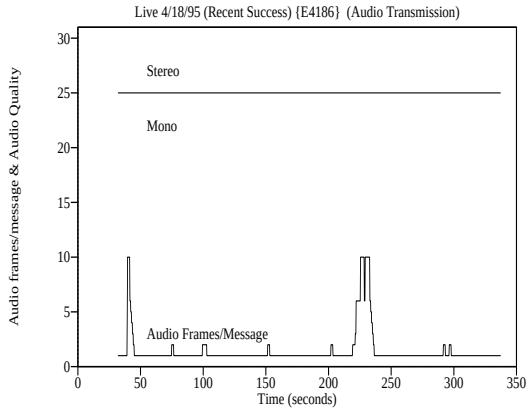
(b) Message Loss



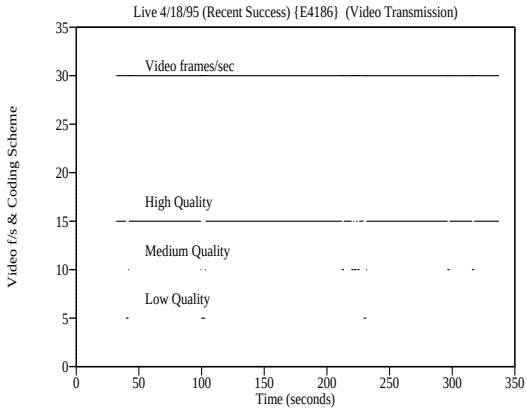
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 6-6: Sitterson Network Experiment - April 18, 1995 - Recent Success

Experiment 4/18	Baseline	VSO	Recent Success
Audio FPS			
Mean	59.39	59.43	59.86
Standard deviation	2.72	3.36	1.42
Minimum	33	21	46
Maximum	65	65	64
Mode/Median	60/60	60/60	60/60
Gaps	163	137	33
Audio Latency: Delivered (Induced)			
Mean	39.19 (17.27)	37.93 (17.52)	37.68 (21.59)
Standard deviation	12.75 (0.62)	13.56 (0.71)	16.27 (13.87)
Minimum	28 (16)	22 (16)	24 (16)
Maximum	132 (18)	136 (19)	130 (96)
Mode/Median	36 (17)/36 (17)	32 (18)/34 (18)	34 (18)/34 (18)
Intervals > 250 ms	0	0	0
Audio Messages			
Frames Sent	18037	18001	18349
Msgs(frames) Lost	141 (141)	114 (114)	6 (6)
Mean Frames Lost	0.47	0.38	0.02
Max Frames Lost	21	24	4
Video FPS			
Mean	29.65	29.69	29.92
Standard deviation	1.49	1.76	0.67
Minimum	17	11	23
Maximum	33	33	33
Mode/Median	30/30	30/30	30/30
Gaps	96	79	18
Video Latency: Delivered			
Mean	103.26	101.28	97.05
Standard deviation	12.24	11.98	4.64
Minimum	94	74	76
Maximum	190	172	122
Mode/Median	98/100	98/98	97/97
Intervals > 250 ms	0	0	0
Video Messages			
Frames Sent	9019	9000	9175
Frames Lost	84	65	8
Mean Frames Lost	0.28	0.22	0.03
Max Frames Lost	11	14	3

Table 6-2: Summary Statistics for April 18 Experiments

6.4.2. Results for Wednesday, April 19, 1995

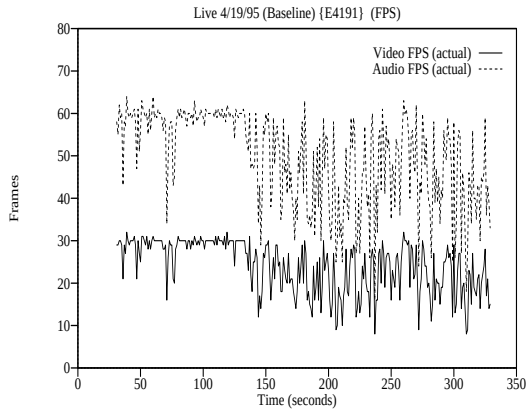
Figures 6-7, 6-8, and 6-9 show the results for conferences using the BL, VSO, and RS algorithms, respectively. We ran the three conferences between 11:00 to 11:30 a.m. on April 19, 1995. Table 6-3 shows the summary statistics for each experiment. In this experiment, there is a heavy traffic load on the Sitterson LAN and this load has a dramatic impact on the quality of the resulting conferences. Figure 6-7 shows that while the BL algorithm worked well with low traffic loads like that experienced on April 18, the algorithm performs poorly when traffic load is high. The delivered frame rates for audio and video deteriorate rapidly with the onset of heavy traffic on the building Ethernet at approximately 150 seconds into the conference (part (a)). Audio quality is consistently poor from 150 seconds to the end of the conference and there are periods when audio is unintelligible. The delivered video frame rates vary widely and often, although the average video rate is 24 frames per second. Audio and video latencies vary frequently and occasionally exceed our 250 *ms* latency guideline (parts (c) and (d)). There is significant message loss (there are over 3,000 audio frames and 1,700 video frames lost during the conference; Table 6-3).

Figure 6-8 shows the results of the conference using VSO. The network is relatively lightly loaded until around 200 seconds into the experiment. VSO adapts to the change in network load by lowering the transmitted quality of the video (part (f)). This illustrates the basic idea behind spatial video scaling, namely reduce the bit rate of the video stream when the network is congested. This strategy results in better delivered frame rates (part (a)) than that delivered by BL. Audio and video frame rates do deteriorate, but not to the extent seen with BL and the perceived quality of the conference is much better. Message loss is lower than with the BL algorithm, but still significant (part (b)).

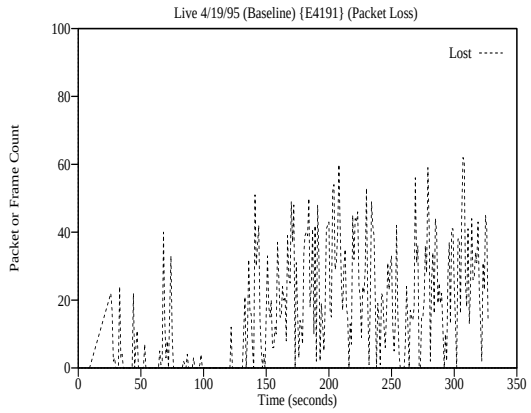
Figure 6-9 shows the results using the RS algorithm. RS reacts to congestion by both adapting the packaging of audio (part (e)) and by adapting both the frame and bit rate of the video stream (part (f)). During the heaviest periods of congestion, RS significantly lowers the transmitted frame rate of video (*e.g.*, at 25-100 and 200-300 seconds into the conference in part (f)) as well as adapting the coding scheme of video. The result of these adaptations is superior audio frame delivery compared with BL and VSO at the expense of video quality during heavy congestion (part (a)). The conference using RS has 244 audio gaps compared with 762 under VSO and 3,104 under BL. Video performance under RS is generally good with degradation only

during the periods of highest network congestion. The adaptations reduce audio and video delivery latencies (parts (c) and (d)) and significantly reduce the number of messages lost (part (b)).

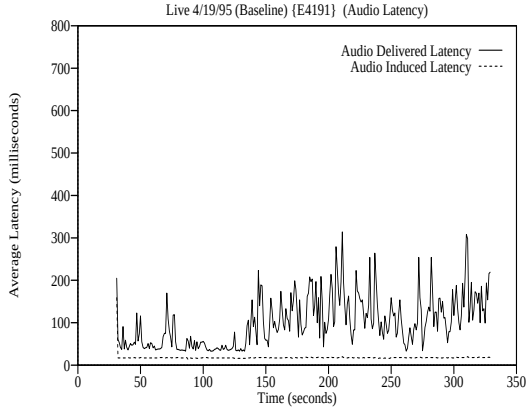
This experiment demonstrates that production networks do experience congestion sufficient to adversely impact conference quality and that non-adaptive transmission schemes like BL are inadequate to preserve conference quality. The experiment also demonstrates that adaptive transmission algorithms can ameliorate the effects of the congestion to some degree. Spatial video scaling can have a degree of success even in heavily congested networks, but RS does a better job of preserving audio performance and also does a better job transmitting only as much data as can be delivered.



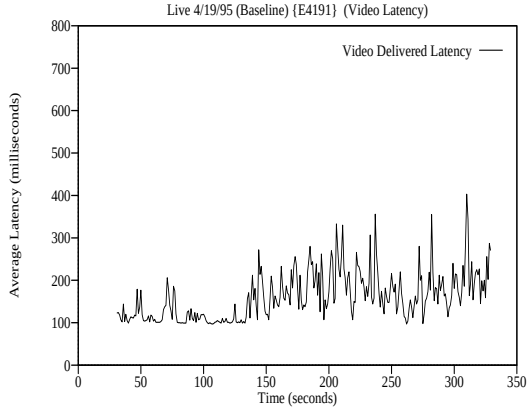
(a) Frames Per Second (FPS)



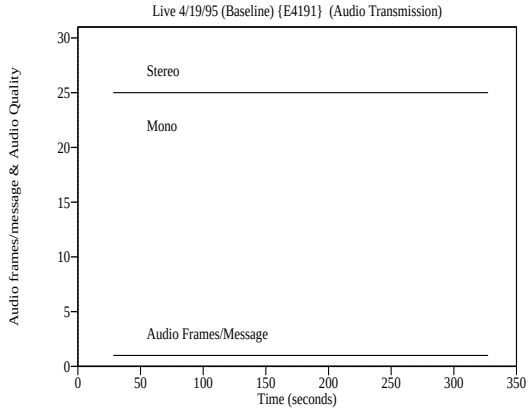
(b) Message Loss



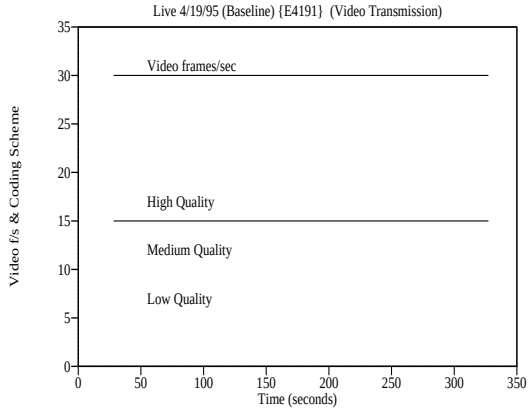
(c) Audio Latency



(d) Video Latency

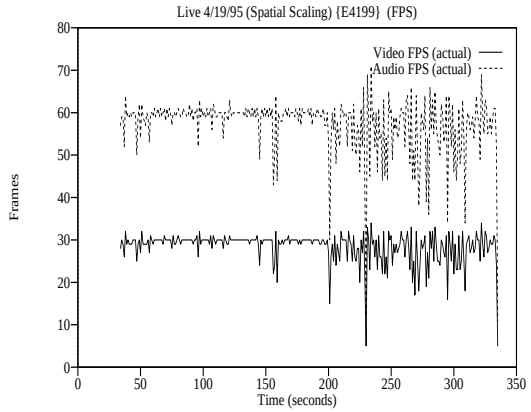


(e) Audio Transmission

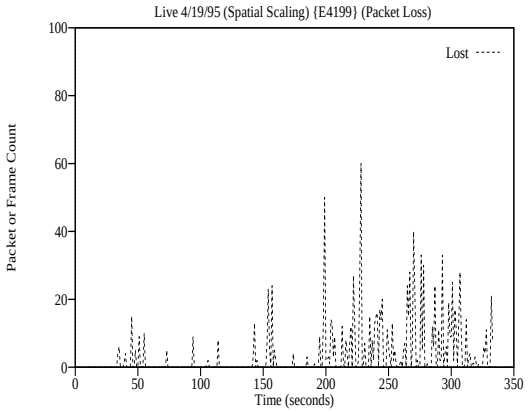


(f) Video Transmission

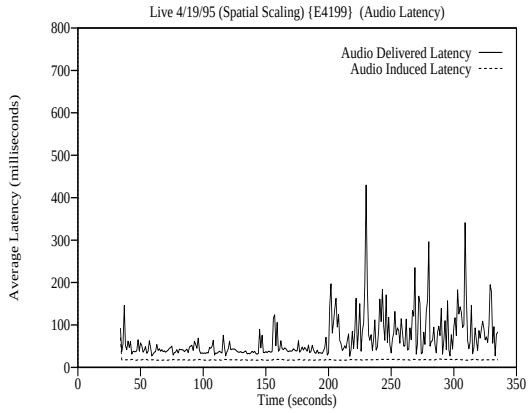
Figure 6-7: Sitterson Network Experiment - April 19, 1995 - Baseline



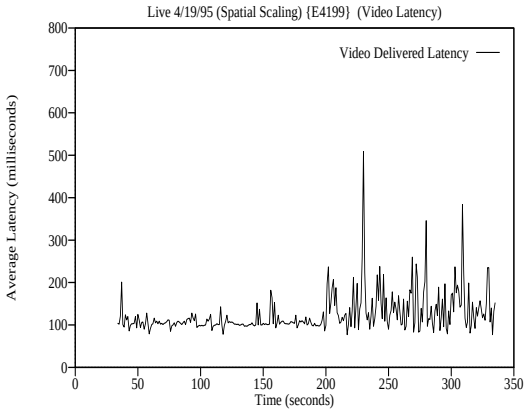
(a) Frames Per Second (FPS)



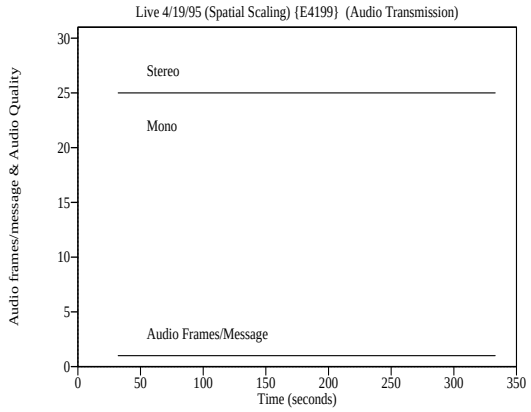
(b) Message Loss



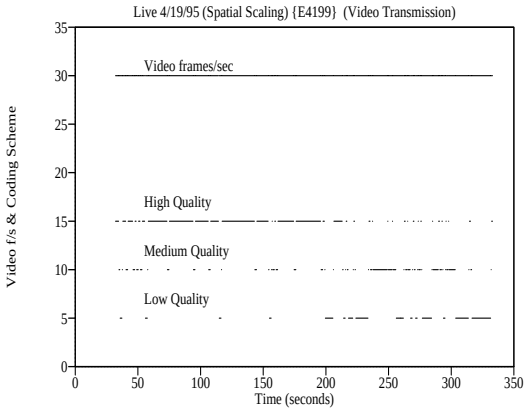
(c) Audio Latency



(d) Video Latency

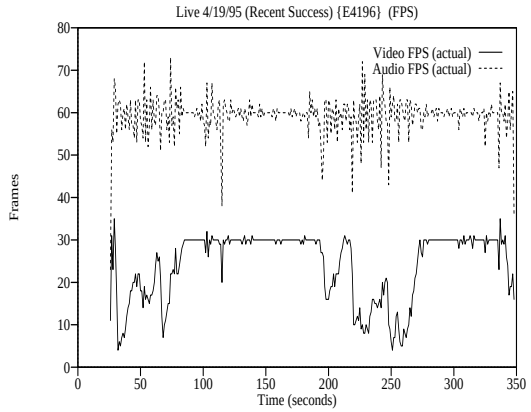


(e) Audio Transmission

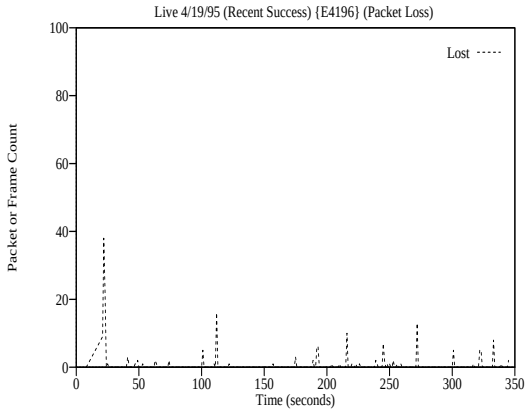


(f) Video Transmission

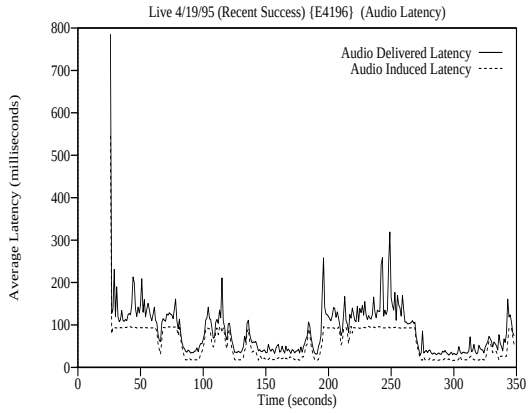
Figure 6-8: Sitterson Network Experiment - April 19, 1995 - Video Scaling Only



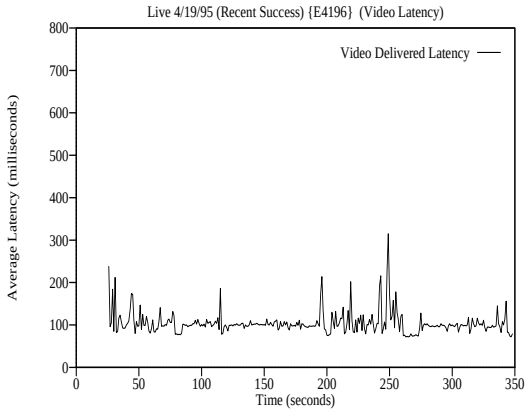
(a) Frames Per Second (FPS)



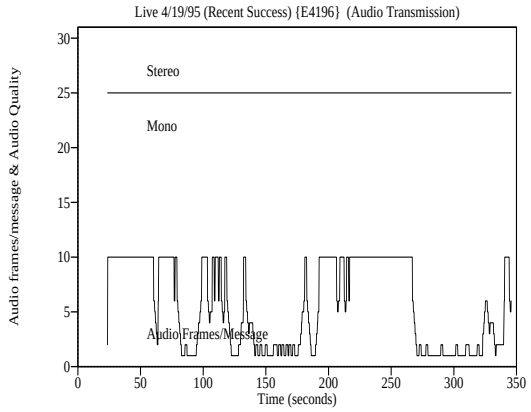
(b) Message Loss



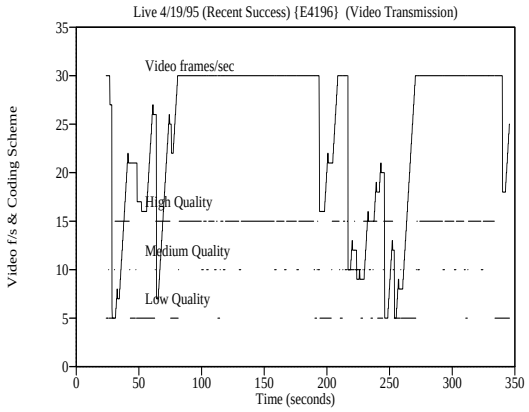
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 6-9: Sitterson Network Experiment - April 19, 1995 - Recent Success

Experiment 4/19	Baseline	VSO	Recent Success
Audio FPS			
Mean	49.66	57.32	59.36
Standard deviation	10.80	6.77	4.24
Minimum	16	11	36
Maximum	64	71	73
Mode/Median	60/54	60/59	60/60
Gaps	3104	762	244
Audio Latency: Delivered (Induced)			
Mean	100.87 (17.46)	66.75 (17.94)	86.43 (56.76)
Standard deviation	59.40 (0.80)	49.30 (0.72)	49.94 (33.74)
Minimum	33 (13)	26 (17)	28 (16)
Maximum	314 (20)	430 (20)	319 (97)
Mode/Median	36 (18)/88 (18)	37 (18)/46 (18)	34 (93)/75 (55)
Intervals > 250 ms	8	3	3
Audio Messages			
Frames Sent	17989	18106	19399
Msgs(frames) Lost	3076 (3076)	750 (750)	90 (207)
Mean Frames Lost	10.22	2.48	0.64
Max Frames Lost	40	40	25
Video FPS			
Mean	24.01	28.50	24.20
Standard deviation	6.08	3.56	7.90
Minimum	8	5	4
Maximum	32	34	35
Mode/Median	30/26	30/30	30/29
Gaps	1788	432	1885
Video Latency: Delivered			
Mean	161.23	125.01	103.85
Standard deviation	56.67	46.61	26.13
Minimum	97	77	72
Maximum	403	509	315
Mode/Median	101/150	101/107	98/99
Intervals > 250 ms	22	5	1
Video Messages			
Frames Sent	8994	9052	7934
Frames Lost	1782	424	108
Mean Frames Lost	5.92	1.40	0.33
Max Frames Lost	23	20	13

Table 6-3: Summary Statistics for April 19 Experiments

6.4.3. Results for Thursday, April 20, 1995

We ran two sets of experiments were on April 20, 1995. Figures 6-10, 6-11, and 6-12 show the results for first set of experiments using the BL, VSO, and RS algorithms, respectively. These conferences were run between about 10:00 to 10:30 a.m. on April 20. Table 6-4 summarizes these results. The results from the first set of experiments are much like those from April 18. The network is lightly loaded and all three transmission techniques perform well, with high delivered frame rates, low latencies, and little message loss. This is particularly true for the BL and RS runs, but towards the end of the VSO conference (see Figure 6-11 (a) at around 200 seconds into the experiment) the network performance appears to be deteriorating. Figure 6-11 (f) shows that starting at around 200 seconds into the experiment, VSO must often lower the quality of the transmitted video. The VSO conference was the last of the experiment run in the first set of experiments and network traffic was building on the building Ethernet.

Figures 6-13, 6-15, and 6-14 show the results for the second set of experiments. These conferences were held between 10:30 a.m. and 11:00 a.m. on April 20. Table 6-5 shows the statistical summaries. Figure 6-13 shows that the performance of the BL conference is much worse than that received only minutes before (see Figure 6-10). Audio and video delivered frame rates are low (Figure 6-13 (a)) and message loss is very high (part (b)). The conference experiences 2,438 audio gaps and 1,425 video gaps (Table 6-5). Latencies vary widely with spikes exceeding the 250 *ms* guideline (parts (c) and (d)).

Congestion on the network continued throughout the following conference using the RS algorithm (Figure 6-15). RS adjusts both the audio packaging and the video scaling trying to adapt to the heavy network congestion, but frequently reaches its adaptation limits. In the case of audio, the operating points in Figure 6-2 prevent more than 10 audio frames from being transmitted in a single message. Figure 6-15 (e) shows that throughout this experiment RS is operating at the maximum available audio packaging. Similarly, the operating points in Figure 6-2 do not allow RS to lower the video frame rate below 5 frames per second. During this experiment RS often reaches the lowest available frame rate for video at the lowest quality coding scheme (*e.g.,* part (f) between 150 and 200 seconds into the conference). The

network congestion is so severe that no operating points available to the RS algorithm can completely ameliorate the effects of the congestion.

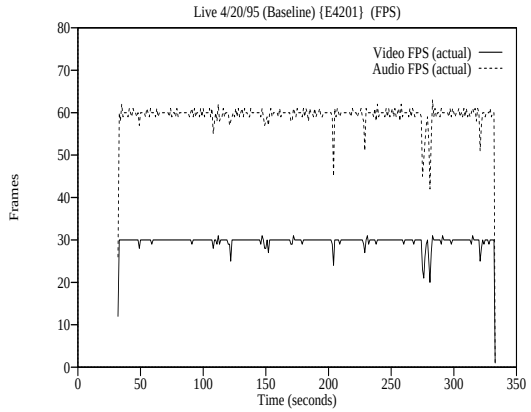
Figure 6-15 (a) shows that the RS adaptations preserve some degree of audio performance at significant expense to the video stream. Even in this extreme environment, RS manages to keep message loss very low and there are significantly fewer audio gaps than with the BL algorithm (486 with RS versus 2,438 with BL; Table 6-5).

Figure 6-14 shows the results of the VSO conference. We ran the VSO conference after the RS experiment. During the early parts of the conference, VSO is sending almost exclusively low quality video (part (f) between 0-150 seconds) and is having trouble delivering consistent audio and video frame rates (part (a)). Like RS, VSO reaches the limits of its available adaptations (*i.e.*, VSO is consistently sending only low quality video), but is unable to completely compensate for the network congestion. At approximately 180 seconds into the conference, network congestion begins to drop and at around 200 seconds into the conference, network conditions are similar to those experienced in the first set of experiments on April 20 (*i.e.*, Figures 6-10, 6-11, and 6-12). VSO is able to transmit high quality video at 30 frames per second (part (f)) and frame delivery is much improved (part (a)).

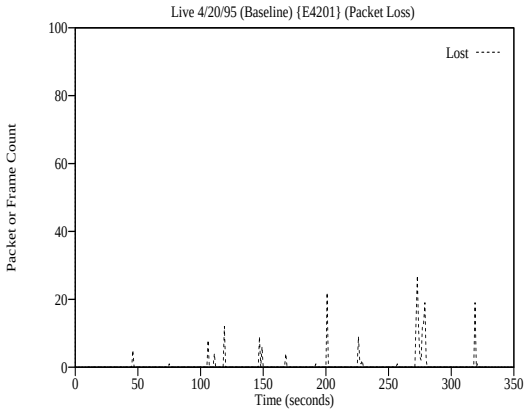
The network was heavily congested throughout the conferences using the BL and RS algorithms; however, the network was congested only over approximately half the conference using the VSO algorithm, so we must be careful comparing the statistics in Table 6-5. For example, the conference using VSO experienced 942 audio gaps while the conference using RS experienced 486 audio gaps. Most of the audio gaps during the VSO experiment occur during the first half of the conference while those during the RS experiment were spread throughout the conference. During the first half of the VSO experiment there are many more gaps than in the RS conference. As a result the perceived quality of the audio during the first half of the VSO conference is much lower than that in the RS conference.

The major conclusion to draw from these set of experiments is that production networks can experience levels of network congestion that are very difficult manage. Both the VSO and RS algorithms reach the limits of their ability to adapt and stay at those limits for long periods of time. No end-to-end adaptive algorithm can provide a quality conference under all network conditions on current LANs. It is possible to

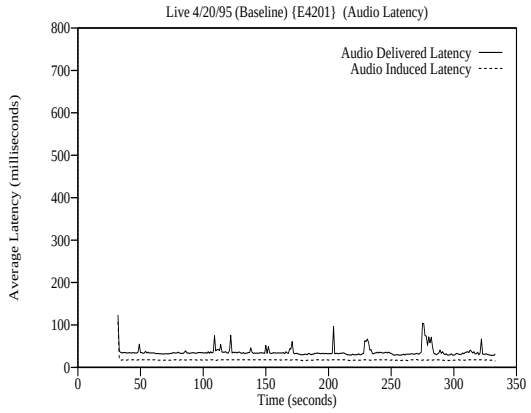
generate enough traffic on these LANs that no best-effort video conferencing system can succeed. Furthermore, production networks experience actual loads that make transmission of a quality video conference difficult or impossible. Network conditions may vary widely over a period of only a few minutes and high network congestion may persist for many minutes. Nevertheless, adaptive algorithms can do much better than non-adaptive techniques at matching the transmitted conference data to the limits of the network environment and can more intelligently select the parts of the media streams delivered. Both VSO and RS deliver significantly higher audio frame rates than BL. Audio is better with RS than with VSO. The BL and VSO algorithms deliver significantly better video frame rates than the RS algorithm, but this is to be expected since BL and VSO never change the transmitted video frame rate and RS intentionally lowers the transmitted video frame rate during congested periods. BL and VSO attempt to force the full video frame rate through the congested network. This results in high message losses (3,834 and 1,431 messages lost, respectively). In contrast, RS attempts to match the transmitted streams to the available network capabilities and loses only 212 messages.



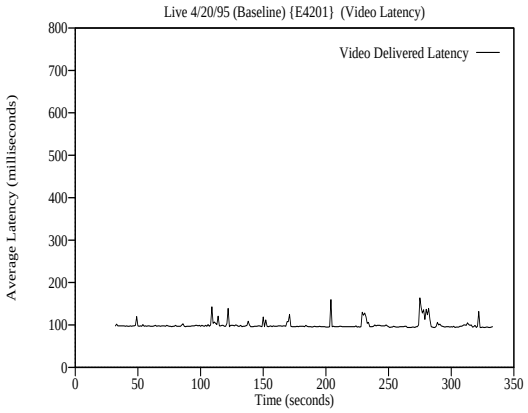
(a) Frames Per Second (FPS)



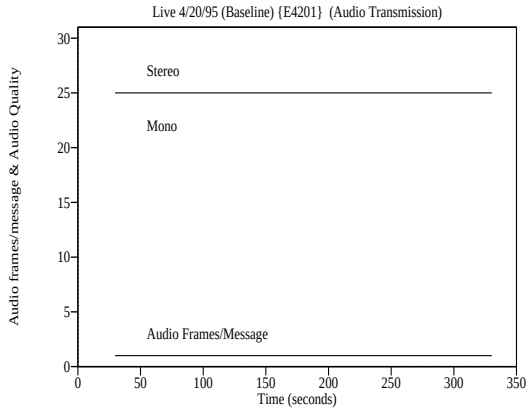
(b) Message Loss



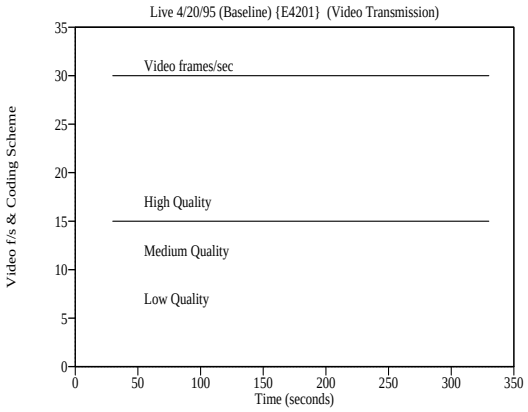
(c) Audio Latency



(d) Video Latency

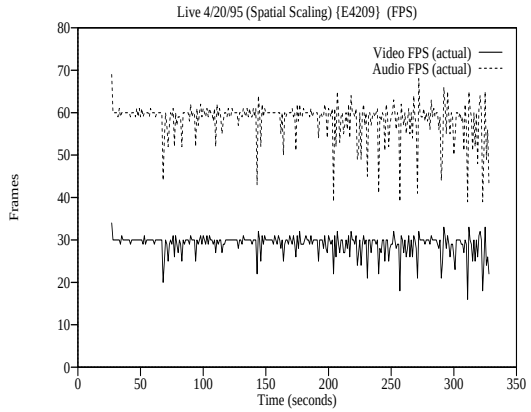


(e) Audio Transmission

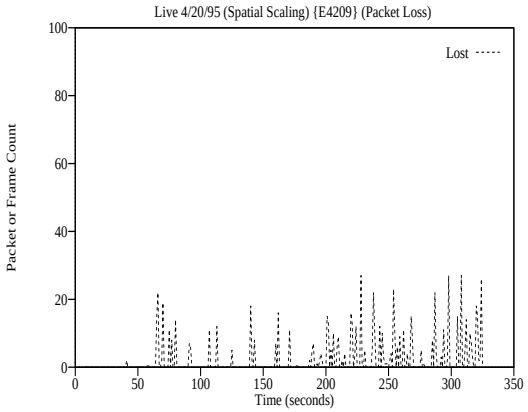


(f) Video Transmission

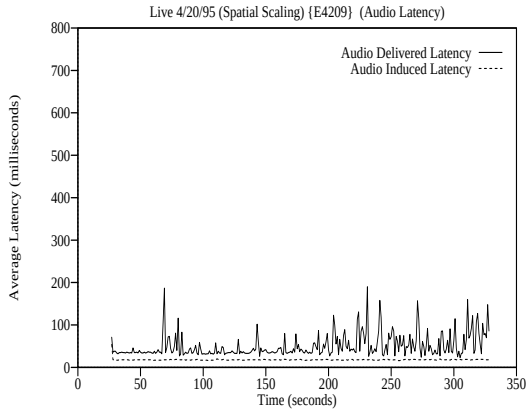
Figure 6-10: Sitterson Network Experiment - April 20, 1995 Set 1 - Baseline



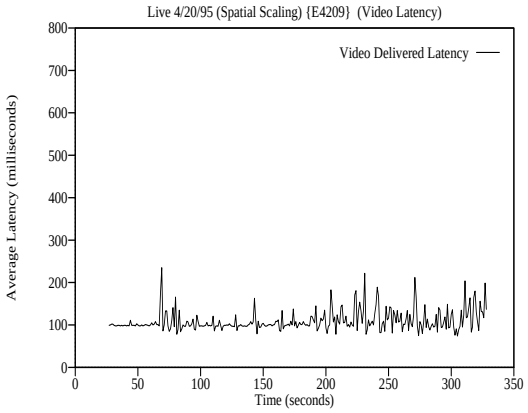
(a) Frames Per Second (FPS)



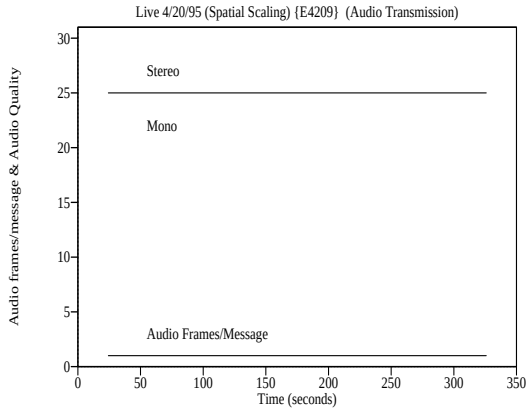
(b) Message Loss



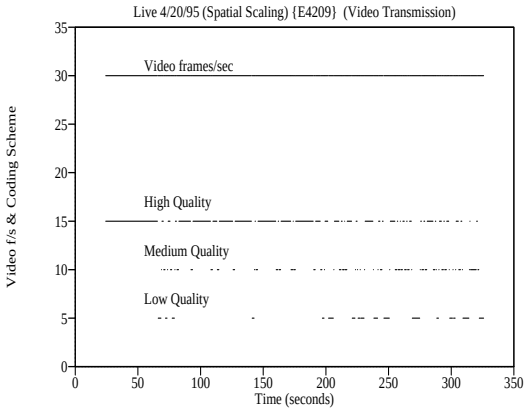
(c) Audio Latency



(d) Video Latency

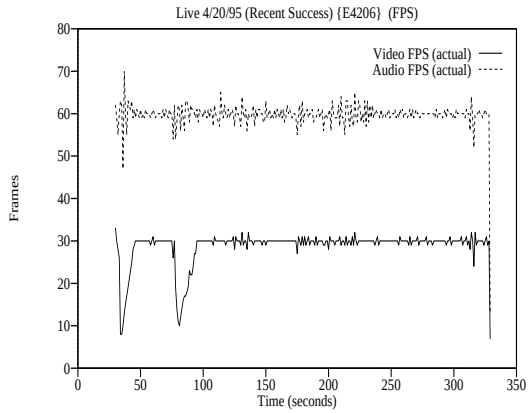


(e) Audio Transmission

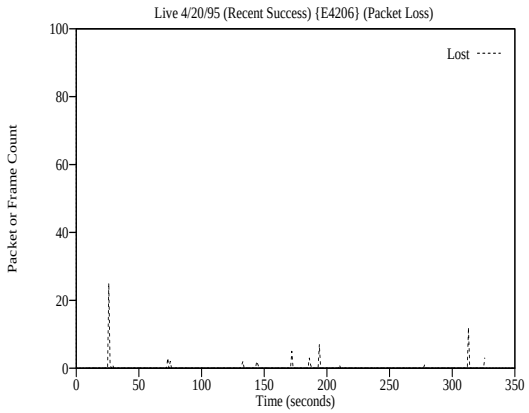


(f) Video Transmission

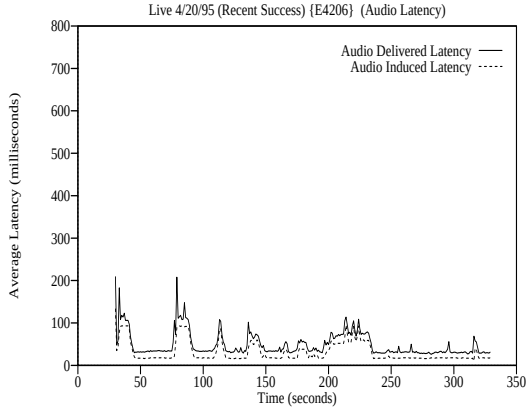
Figure 6-11: Sitterson Network Experiment - April 20, 1995 Set 1 - Video Scaling Only



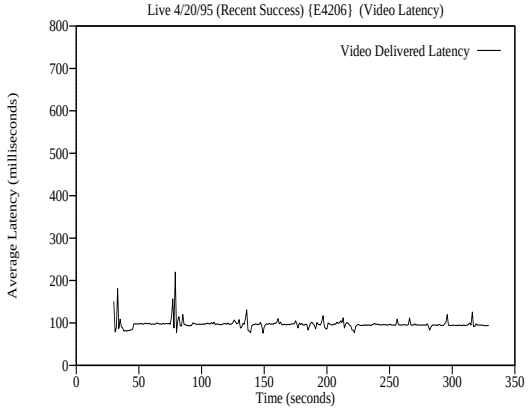
(a) Frames Per Second (FPS)



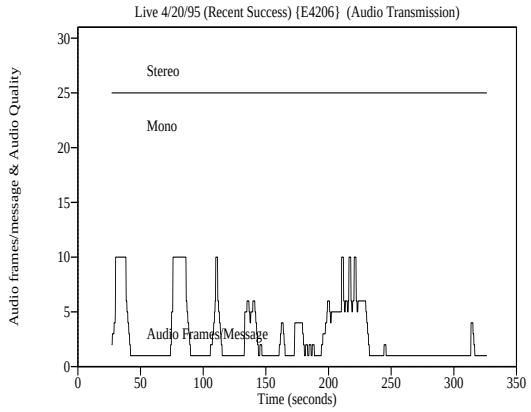
(b) Message Loss



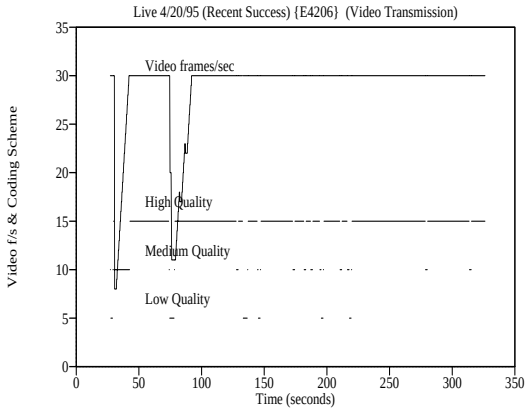
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission

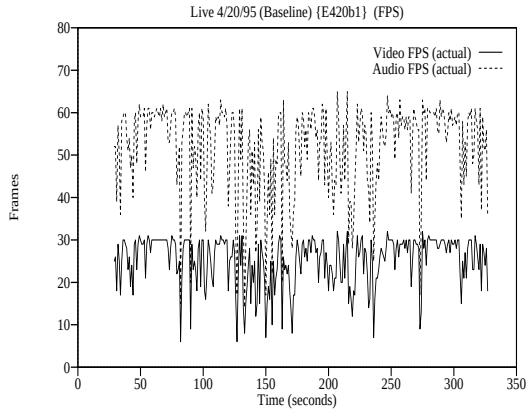


(f) Video Transmission

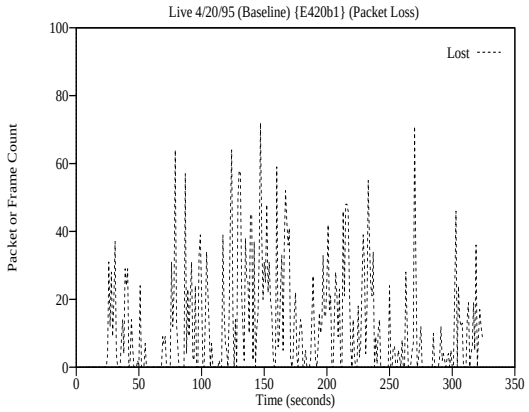
Figure 6-12: Sitterson Network Experiment - April 20, 1995 Set 1 - Recent Success

Experiment 4/20	Baseline	VSO	Recent Success
Audio FPS			
Mean	59.29	58.32	59.68
Standard deviation	4.00	4.45	3.34
Minimum	1	39	13
Maximum	63	68	70
Mode/Median	60/60	60/60	60/60
Gaps	156	494	70
Audio Latency: Delivered (Induced)			
Mean	35.91 (17.56)	50.45 (18.00)	48.54 (31.26)
Standard deviation	10.30 (0.61)	27.61 (0.81)	27.58 (22.93)
Minimum	28 (15)	23 (16)	26 (10)
Maximum	104 (19)	190 (20)	208 (95)
Mode/Median	34 (18)/34 (18)	34 (18)/38 (18)	31 (17)/34 (18)
Intervals > 250 ms	0	0	0
Audio Messages			
Frames Sent	18046	18155	17990
Msgs(frames) Lost	135 (135)	463 (463)	42 (49)
Mean Frames Lost	0.45	1.53	0.16
Max Frames Lost	17	17	16
Video FPS			
Mean	29.62	29.00	28.67
Standard deviation	2.03	2.41	4.20
Minimum	1	16	7
Maximum	31	33	32
Mode/Median	30/30	30/30	30/30
Gaps	87	300	384
Video Latency: Delivered			
Mean	99.75	109.43	97.13
Standard deviation	9.61	24.68	11.38
Minimum	94	74	76
Maximum	164	235	220
Mode/Median	97/97	98/100	95/96
Intervals > 250 ms	0	0	0
Video Messages			
Frames Sent	9023	9079	8648
Frames Lost	76	285	29
Mean Frames Lost	0.25	0.94	0.10
Max Frames Lost	10	11	9

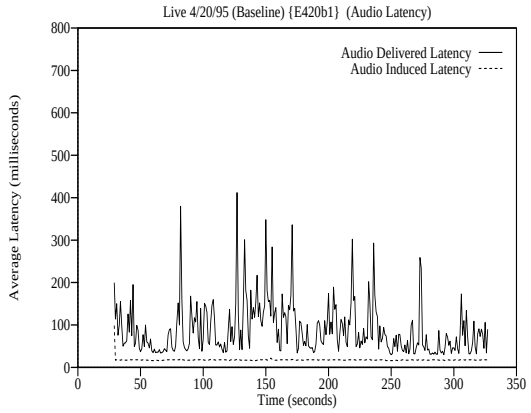
Table 6-4: Summary Statistics for April 20 (set 1) Experiments



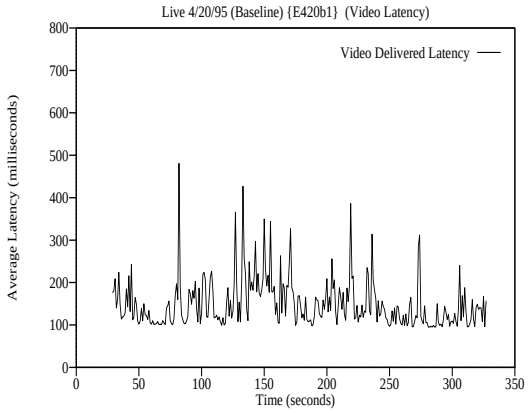
(a) Frames Per Second (FPS)



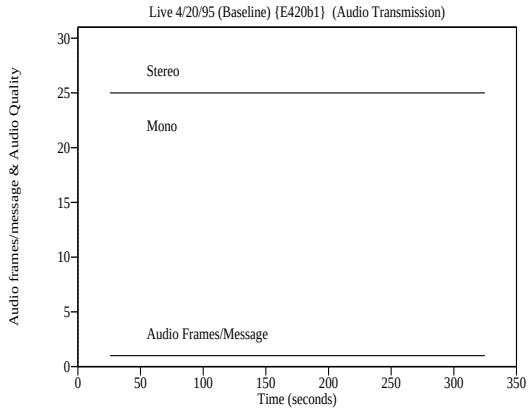
(b) Message Loss



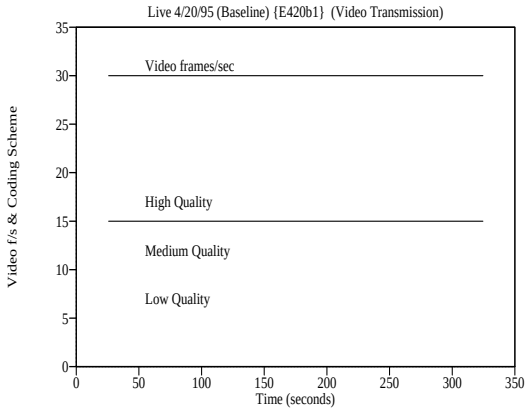
(c) Audio Latency



(d) Video Latency

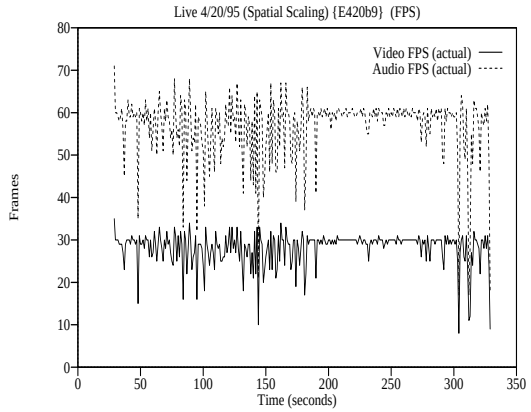


(e) Audio Transmission

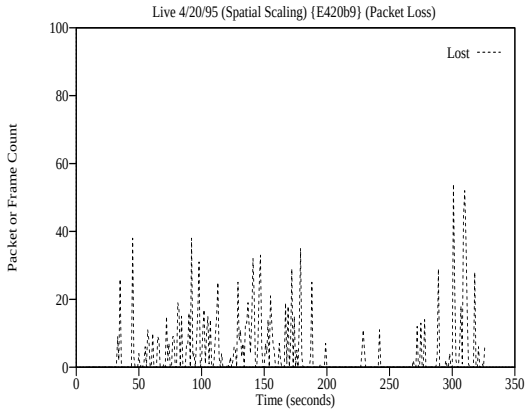


(f) Video Transmission

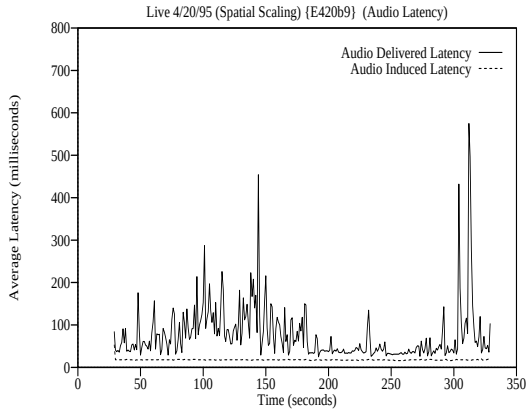
Figure 6-13: Sitterson Network Experiment - April 20, 1995 Set 2 - Baseline



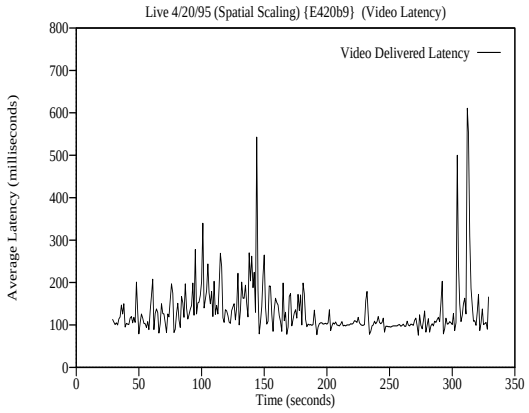
(a) Frames Per Second (FPS)



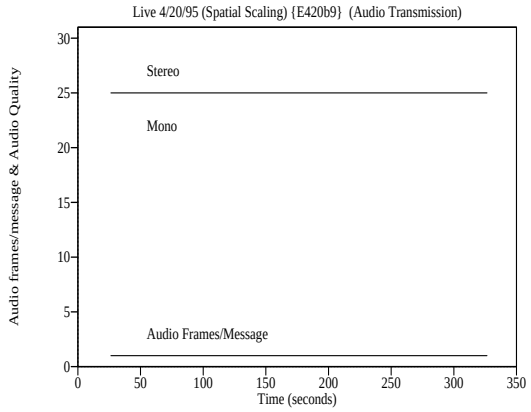
(b) Message Loss



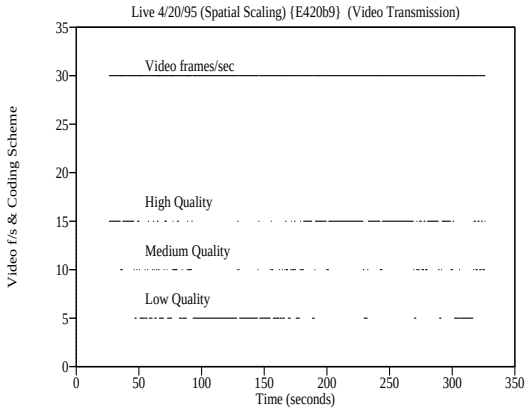
(c) Audio Latency



(d) Video Latency

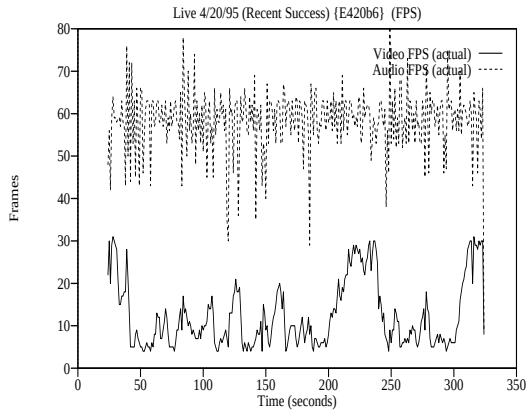


(e) Audio Transmission

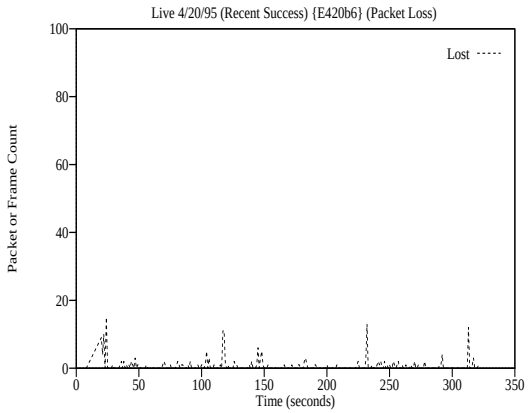


(f) Video Transmission

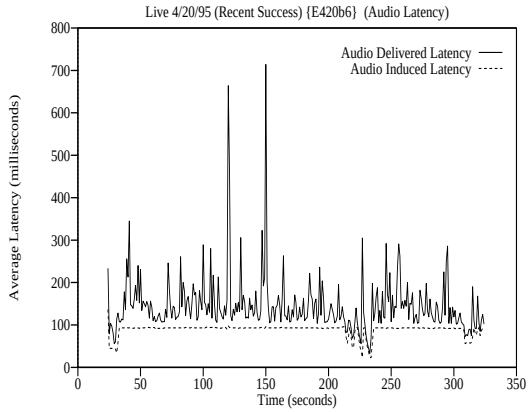
Figure 6-14: Sitterson Network Experiment - April 20, 1995 Set 2 - Video Scaling Only



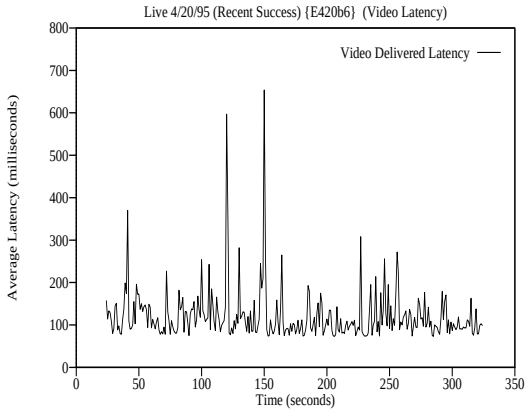
(a) Frames Per Second (FPS)



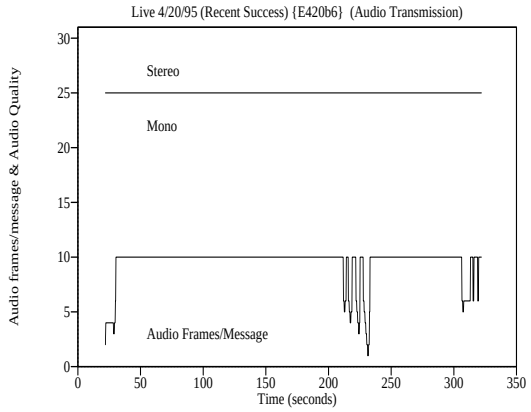
(b) Message Loss



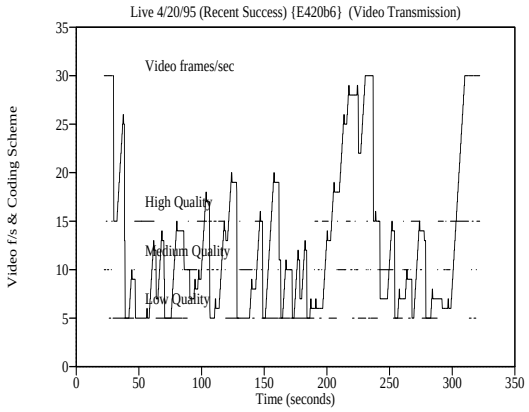
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 6-15: Sitterson Network Experiment - April 20, 1995 Set 2 - Recent Success

Experiment 4/20	Baseline	VSO	Recent Success
Audio FPS			
Mean	51.85	56.71	58.29
Standard deviation	10.44	7.57	7.73
Minimum	14	17	9
Maximum	65	68	82
Mode/Median	60/56	60/59	63/59
Gaps	2438	942	486
Audio Latency: Delivered (Induced)			
Mean	88.70 (17.56)	76.65 (17.84)	144.45 (88.41)
Standard deviation	60.67 (0.77)	66.81 (0.75)	67.90 (13.60)
Minimum	30 (16)	25 (15)	33 (23)
Maximum	412 (22)	575 (20)	714 (98)
Mode/Median	34 (18)/72 (18)	35 (18)/55 (18)	104 (93)/129 (93)
Intervals > 250 ms	9	5	17
Audio Messages			
Frames Sent	17953	18070	18038
Msgs(frames) Lost	2420 (2420)	918 (918)	67 (450)
Mean Frames Lost	8.07	3.04	1.50
Max Frames Lost	47	36	30
Video FPS			
Mean	25.24	28.18	12.43
Standard deviation	5.78	3.91	7.79
Minimum	6	8	4
Maximum	32	34	31
Mode/Median	30/27	30/30	6/9
Gaps	1425	527	5262
Video Latency: Delivered			
Mean	149.41	132.84	120.74
Standard deviation	57.12	64.40	61.62
Minimum	95	76	73
Maximum	481	611	654
Mode/Median	101/131	98/110	99/103
Intervals > 250 ms	14	11	11
Video Messages			
Frames Sent	8976	9035	3918
Frames Lost	1414	513	145
Mean Frames Lost	4.71	1.70	0.48
Max Frames Lost	25	18	11

Table 6-5: Summary Statistics for April 20 (set 2) Experiments

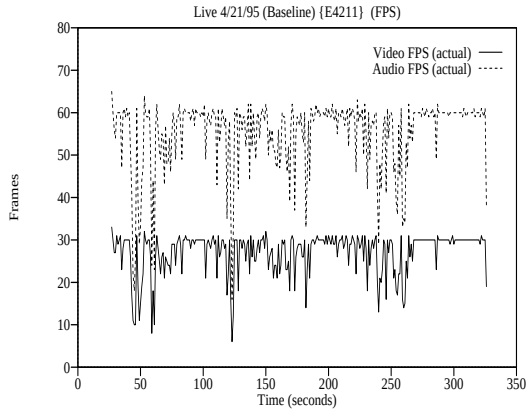
6.4.4. Results for Friday, April 21, 1995

Figures 6-16, 6-17, and 6-18 show the results for conferences using the BL, VSO, and RS algorithms, respectively. We held these conferences between 10:00 to 10:30 a.m. on April 21, 1995. Table 6-6 gives the summary statistics for the experiment. Figure 6-16 (a) and (b) shows the effect of moderate congestion on the delivered audio and video frame rates and loss rates for the BL algorithm. Both audio and video latency vary greatly during the course of the conference with several intervals over 250 *ms* (parts (c) and (d)). There are over 1,400 audio gaps (Table 6-6).

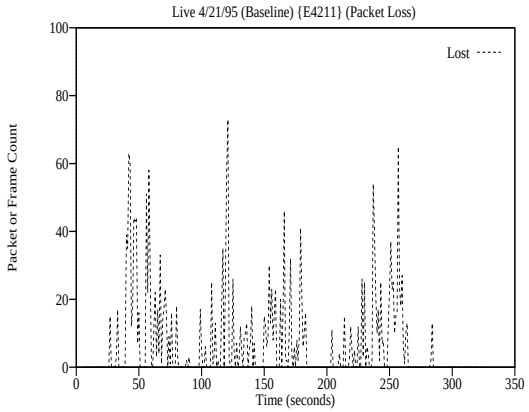
During the RS conference (Figure 6-18), congestion remains at an easily manageable level until approximately 220 milliseconds into the conference. By adapting the audio packaging, video message rate, and video bit rate (parts (e) and (f)), RS is able to maintain audio performance with degraded video performance during the congested period (part (a)). Latency and message loss are low for both the audio and video streams (parts (b), (c), and (d)).

In contrast, we ran the VSO conference (Figure 6-17) just after the completion of the RS conference and VSO is unable to deal with the high congestion levels. Audio and video frame rates are inconsistent (part (a)) and there are many audio gaps (over 1,700 compared to 125 with RS). The audio and video streams have periods of very high latency (parts (c) and (d)) and message loss is high (part (b)). VSO attempts to address the congestion by lowering the quality of the transmitted video (part (f)), but scaling alone proves insufficient to maintain adequate conference quality.

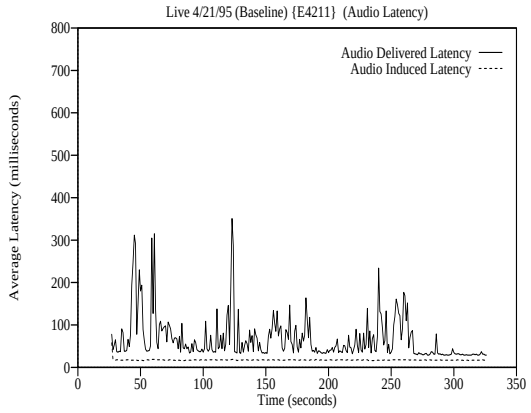
The conclusion from the April 20 and April 21 experiments is that RS is better at adapting to heavy network congestion than VSO or BL. We demonstrated this in the controlled network experiments (see Chapter 5) and the last two experiments demonstrate this on a production network. RS maintains the quality of the audio stream and limits message loss even under heavy network congestion (on April 20 set 2). In a similar environment (on April 21), the VSO algorithm does not deliver good audio and message loss is high. VSO delivers better video frame rates than RS, but video is less important than audio for perceived conference quality and audio should be favored over video when a choice must be made between the two streams.



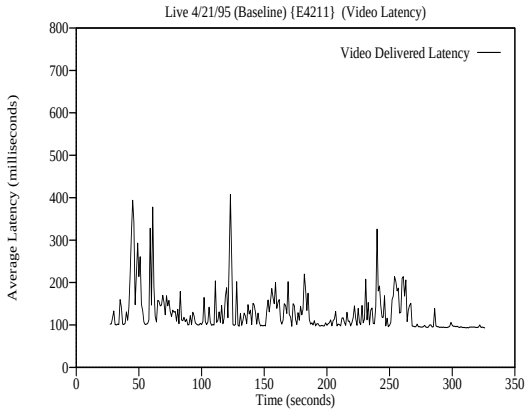
(a) Frames Per Second (FPS)



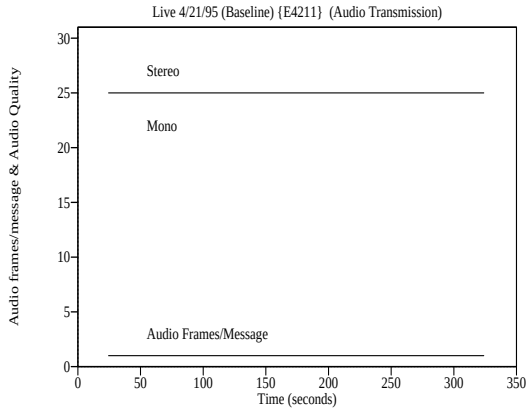
(b) Message Loss



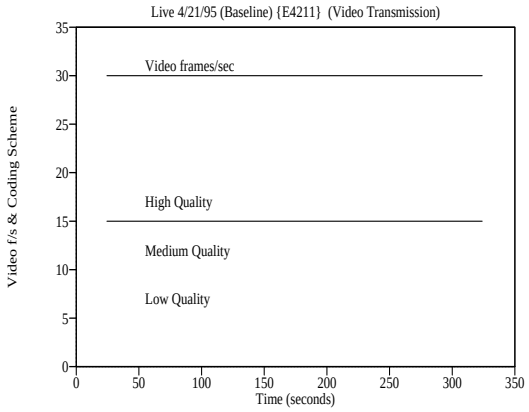
(c) Audio Latency



(d) Video Latency

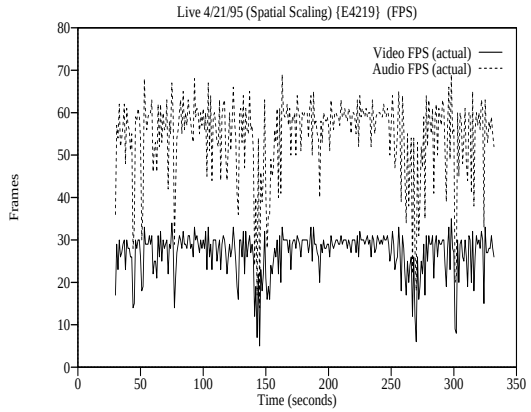


(e) Audio Transmission

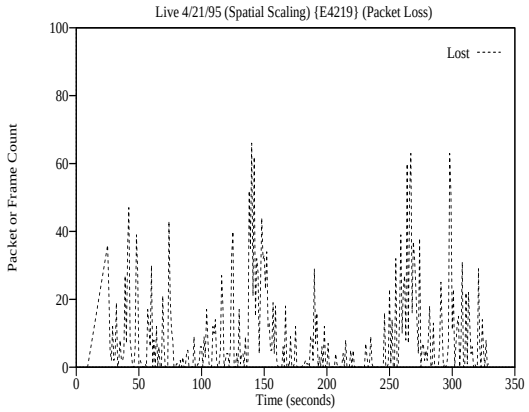


(f) Video Transmission

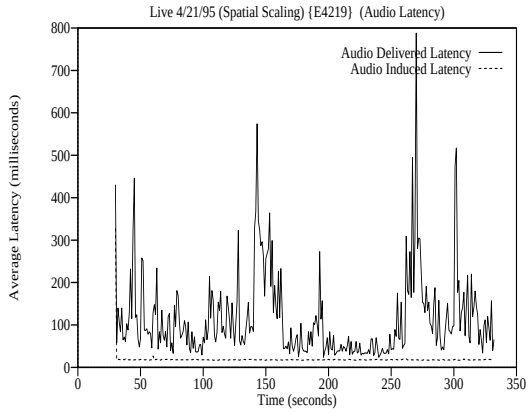
Figure 6-16: Sitterson Network Experiment - April 21, 1995 - Baseline



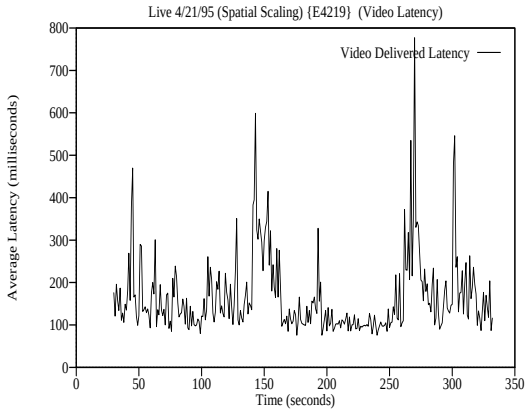
(a) Frames Per Second (FPS)



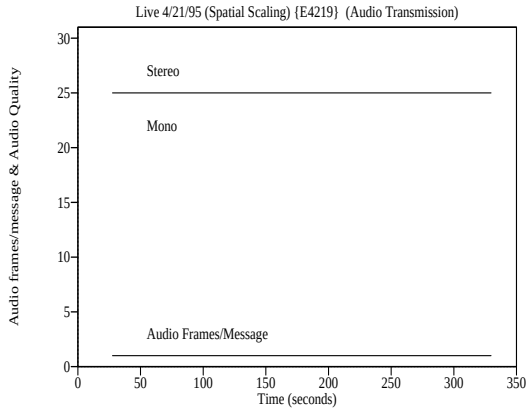
(b) Message Loss



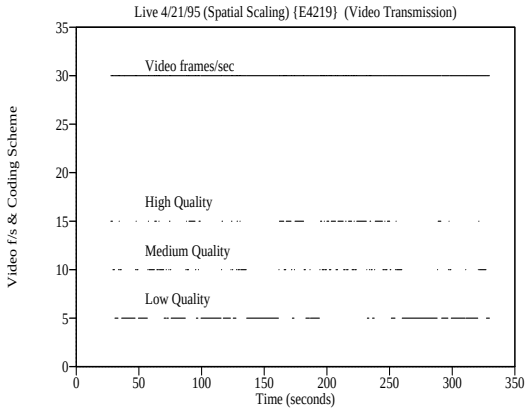
(c) Audio Latency



(d) Video Latency

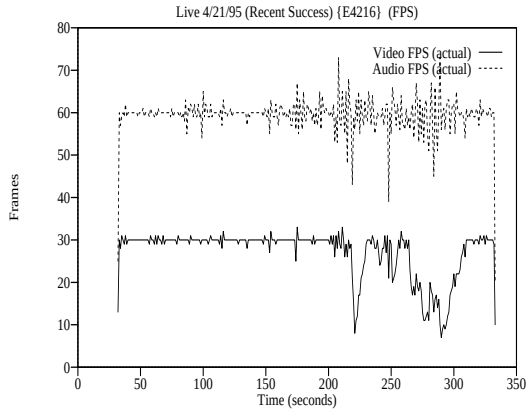


(e) Audio Transmission

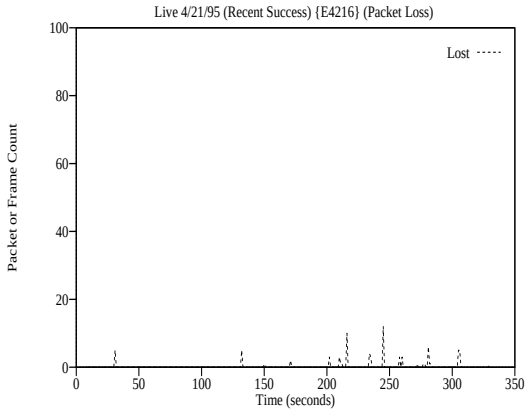


(f) Video Transmission

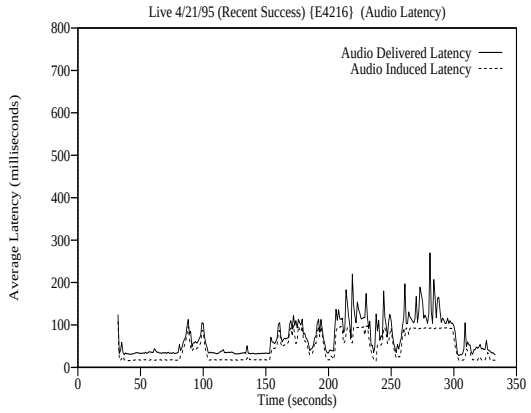
Figure 6-17: Sitterson Network Experiment - April 21, 1995 - Video Scaling Only



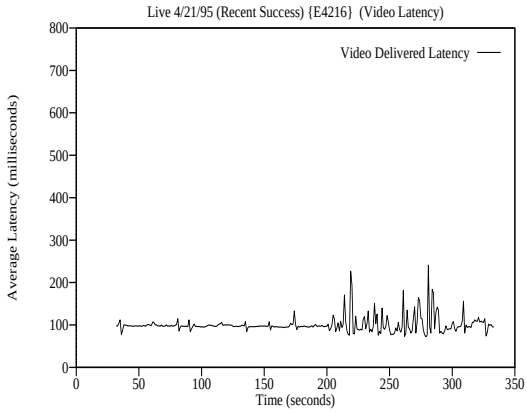
(a) Frames Per Second (FPS)



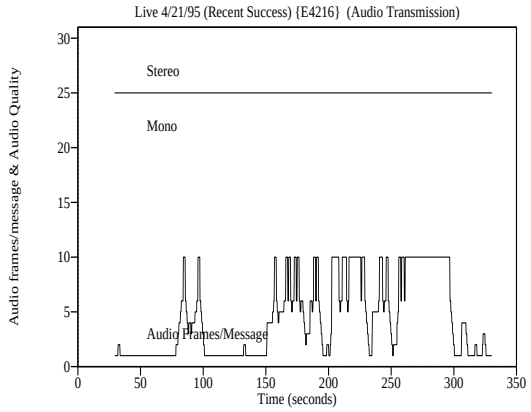
(b) Message Loss



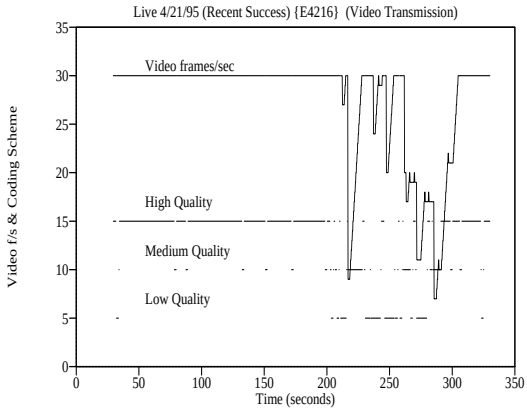
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 6-18: Sitterson Network Experiment - April 21, 1995 - Recent Success

Experiment 4/21	Baseline	VSO	Recent Success
Audio FPS			
Mean	55.14	54.41	59.48
Standard deviation	8.54	9.65	4.04
Minimum	16	14	20
Maximum	64	69	73
Mode/Median	60/59	60/57	60/60
Gaps	1434	1706	125
Audio Latency: Delivered (Induced)			
Mean	67.08 (17.36)	113.49 (18.04)	70.62 (46.84)
Standard deviation	52.78 (0.67)	97.94 (0.87)	42.58 (31.13)
Minimum	29 (16)	24 (16)	28 (15)
Maximum	351 (19)	788 (27)	270 (99)
Mode/Median	29 (17)/46 (17)	38 (18)/84 (18)	33 (18)/56 (36)
Intervals > 250 ms	5	29	1
Audio Messages			
Frames Sent	17977	18205	18043
Msgs(frames) Lost	1413 (1413)	1676 (1676)	25 (97)
Mean Frames Lost	4.71	5.51	0.32
Max Frames Lost	48	42	18
Video FPS			
Mean	27.10	26.95	27.20
Standard deviation	4.93	5.02	5.64
Minimum	6	5	7
Maximum	32	35	32
Mode/Median	30/30	30/29	30/30
Gaps	858	934	826
Video Latency: Delivered			
Mean	129.18	164.75	100.55
Standard deviation	49.70	91.28	19.95
Minimum	93	76	72
Maximum	408	777	241
Mode	95	98	97
Median	109	134	97
Video Messages			
Frames Sent	8988	9102	8262
Frames Lost	846	919	53
Mean Frames Lost	2.82	3.02	0.18
Max Frames Lost	25	25	9

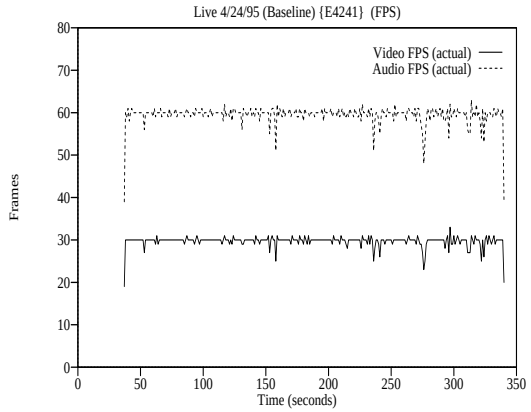
Table 6-6: Summary Statistics for April 21 Experiments

6.4.5. Results for Monday, April 24, 1995

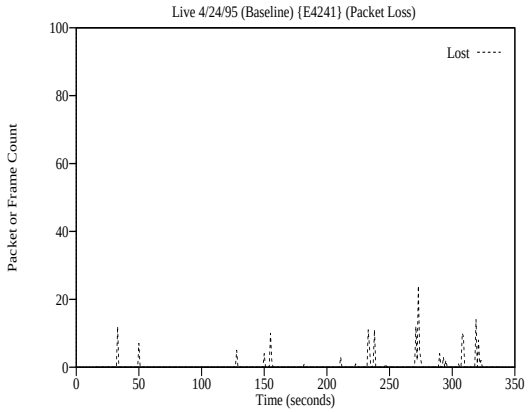
Figures 6-19, 6-20, and 6-21 show the results for conferences using BL, VSO, and RS that were held between 11:00 to 11:30 a.m. on April 24, 1995. Table 6-7 shows the statistical summary. These results are much like those from April 18 and demonstrate that RS imposes no performance penalty when network congestion is low. Like the experiment on April 18, even with essentially no competing traffic, the RS algorithm produces a slightly better conference with less loss and higher delivered audio frame rates than the BL and VSO algorithms.

6.4.6. Results for Tuesday, April 25, 1995

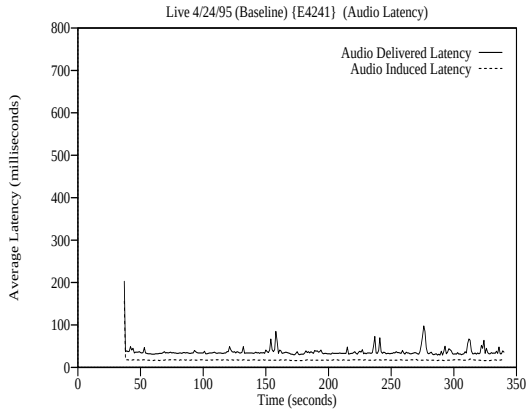
Figures 6-22, 6-23, and 6-24 show the results for conferences using the BL, VSO, and RS algorithms. We held these conferences between 10:30 to 11:00 a.m. on April 25, 1995. Table 6-8 summarizes the results. There is light to moderate network traffic during most of these conferences. Figure 6-23 shows that VSO produces better overall results than BL and better video results than RS. Figure 6-24 (e) and (f) show that around 240 seconds into the conference RS aggressively reacts to what appears to be building network congestion. RS raises the audio packaging rates and lowers the video frame rates (lowering both message rates). Although this leads to a drastic short-term reduction in delivered video frame rates, we believe aggressive responses to congestion are warranted to prevent conditions from getting worse and to control the message loss of the conference. When the congestion disappears, RS quickly returns to previous transmission levels (Figure 6-24 (f) at about 260 seconds into the conference). The conclusion from this experiment is that under light, sporadic congestion RS sometimes gives inferior video performance when compared with VSO. However, given the potential impact of slow reaction to congestion, we believe the benefits of an aggressive, multi-dimensional response to congestion outweigh the impact of a short-term degradation in the video stream. In the case of this particular set of experiments, small scaling adjustments were enough to address the congestion, but this is not always the case.



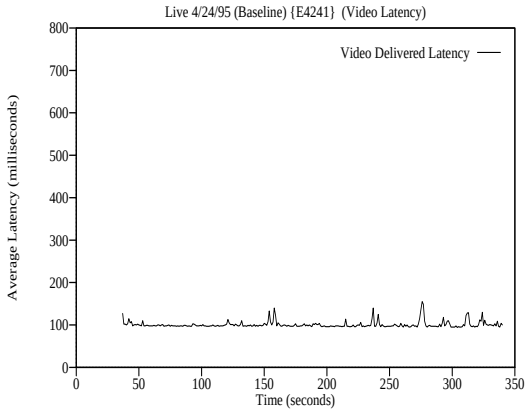
(a) Frames Per Second (FPS)



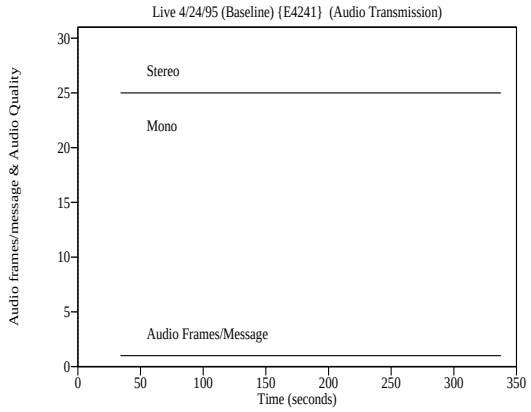
(b) Message Loss



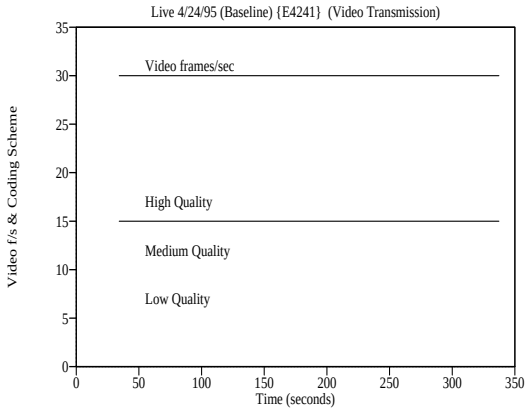
(c) Audio Latency



(d) Video Latency

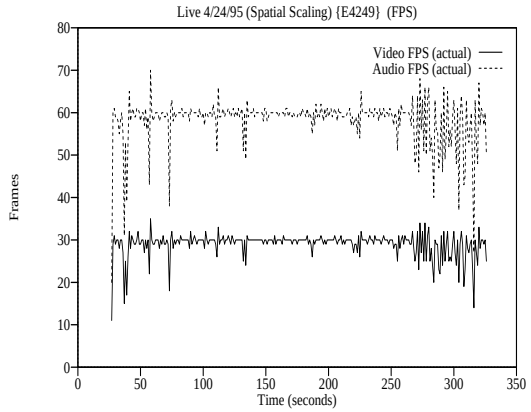


(e) Audio Transmission

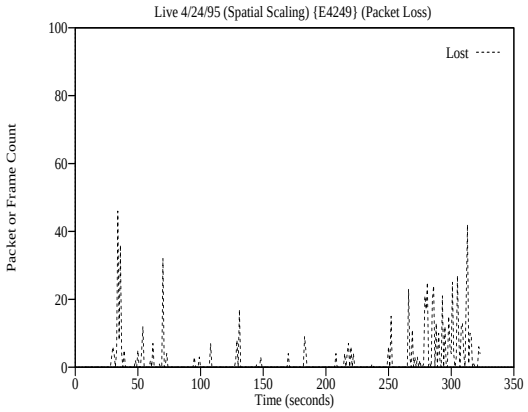


(f) Video Transmission

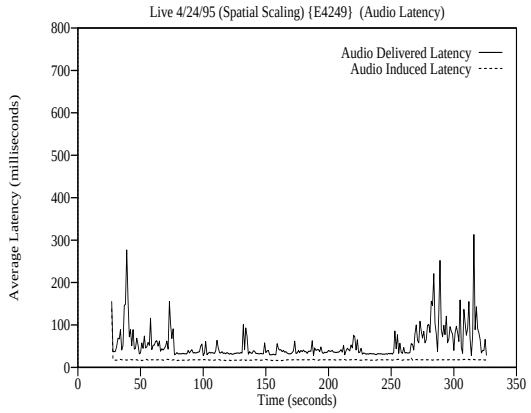
Figure 6-19: Sitterson Network Experiment - April 24, 1995 - Baseline



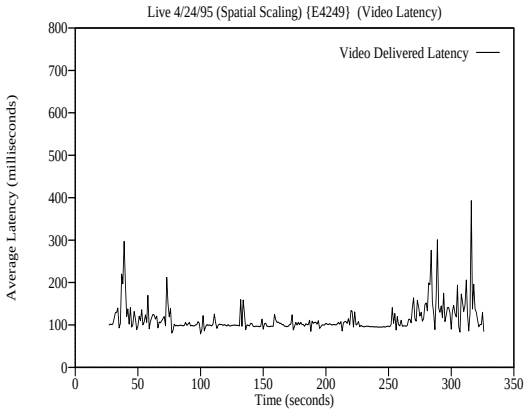
(a) Frames Per Second (FPS)



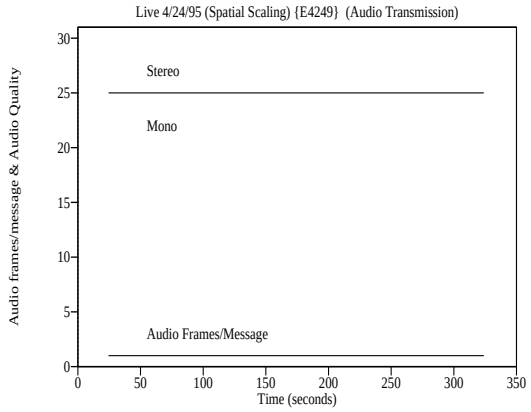
(b) Message Loss



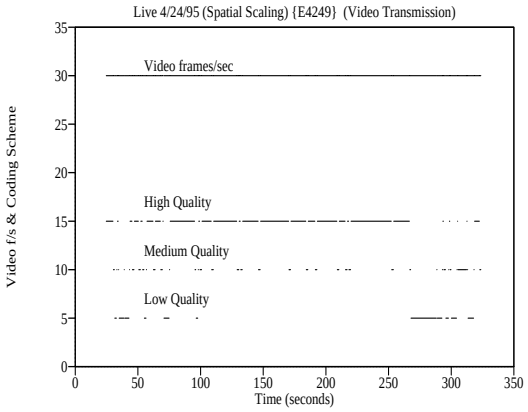
(c) Audio Latency



(d) Video Latency

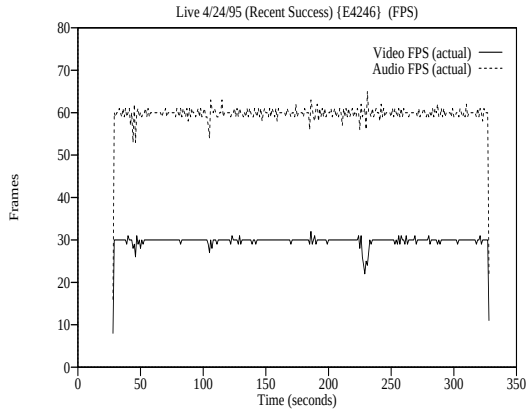


(e) Audio Transmission

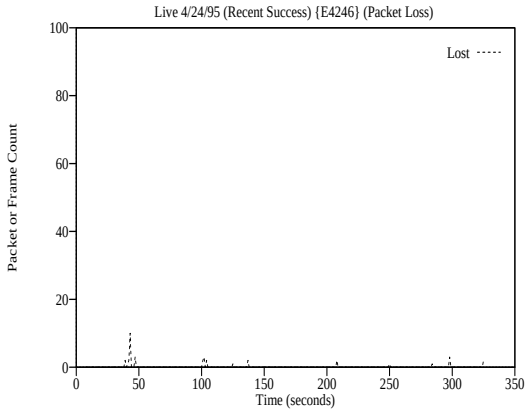


(f) Video Transmission

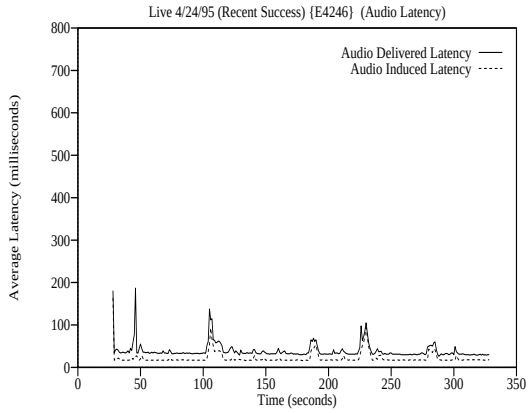
Figure 6-20: Sitterson Network Experiment - April 24, 1995 - Video Scaling Only



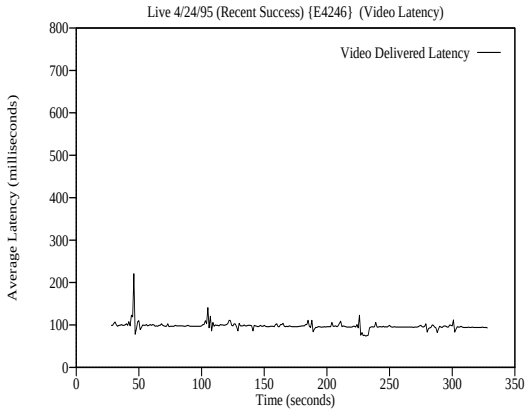
(a) Frames Per Second (FPS)



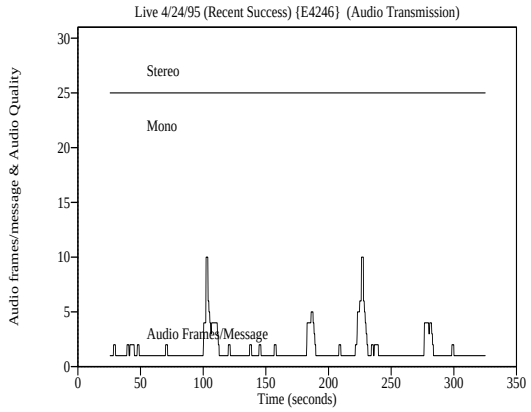
(b) Message Loss



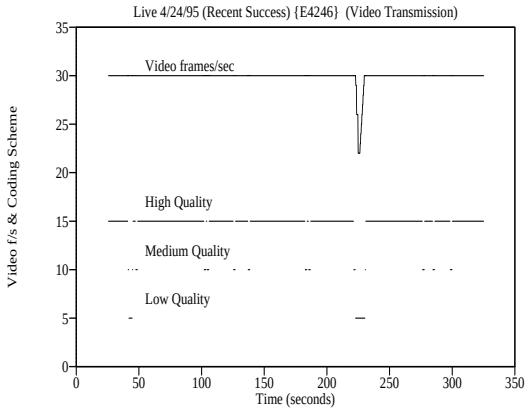
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission

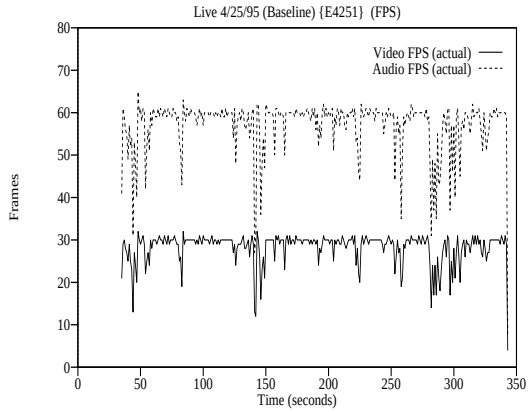


(f) Video Transmission

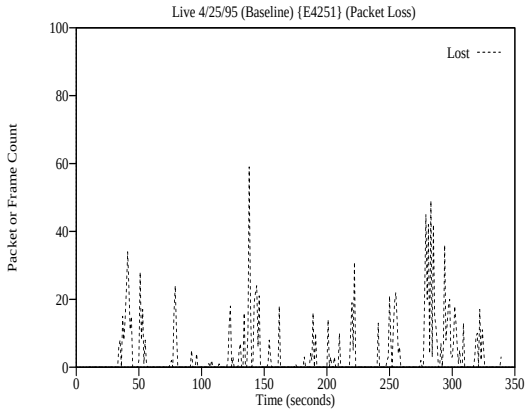
Figure 6-21: Sitterson Network Experiment - April 24, 1995 - Recent Success

Experiment 4/24	Baseline	VSO	Recent Success
Audio FPS			
Mean	59.53	58.33	59.73
Standard deviation	2.04	5.04	2.51
Minimum	39	27	22
Maximum	63	70	65
Mode/Median	60/60	60/60	60/60
Gaps	129	494	42
Audio Latency: Delivered (Induced)			
Mean	36.44 (17.29)	54.05 (17.78)	38.37 (21.41)
Standard deviation	8.54 (0.64)	37.27 (0.80)	16.27 (10.90)
Minimum	29 (16)	28 (16)	26 (16)
Maximum	98 (20)	313 (23)	187 (94)
Mode/Median	34 (17)/34 (17)	32 (18)/39 (18)	31 (17)/33 (17)
Intervals > 250 ms	0	3	0
Audio Messages			
Frames Sent	18237	17960	17985
Msgs(frames) Lost	108 (108)	471 (471)	16 (24)
Mean Frames Lost	0.35	1.57	0.08
Max Frames Lost	15	30	10
Video FPS			
Mean	29.73	29.09	29.73
Standard deviation	1.17	2.64	1.43
Minimum	20	14	11
Maximum	33	35	32
Mode/Median	30/30	30/30	30/30
Gaps	79	272	65
Video Latency: Delivered			
Mean	100.73	114.07	97.58
Standard deviation	8.02	33.88	9.68
Minimum	94	79	74
Maximum	155	393	221
Mode/Median	98/98	97/102	95/97
Intervals > 250 ms	0	4	0
Video Messages			
Frames Sent	9118	8981	8960
Frames Lost	66	260	21
Mean Frames Lost	0.22	0.87	0.07
Max Frames Lost	9	16	5

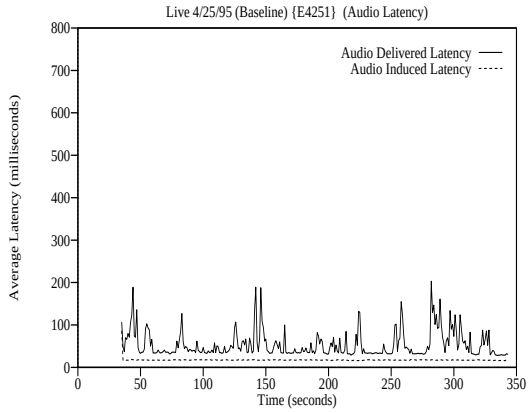
Table 6-7: Summary Statistics for April 24 Experiments



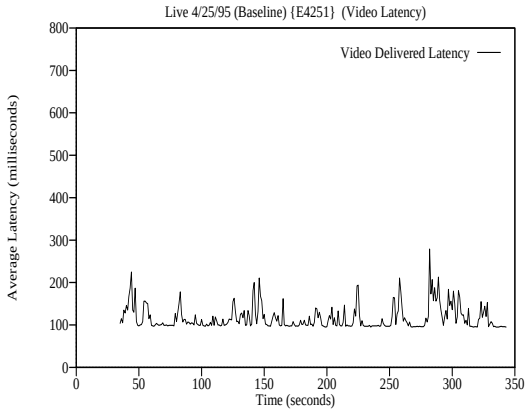
(a) Frames Per Second (FPS)



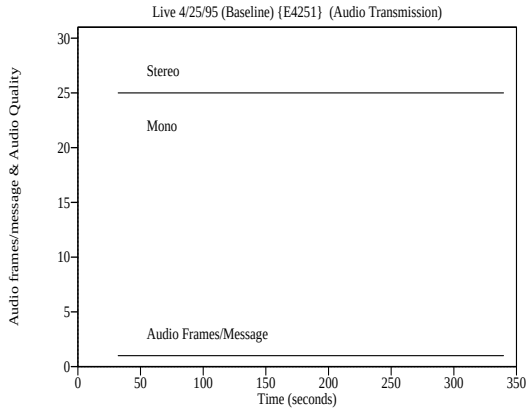
(b) Message Loss



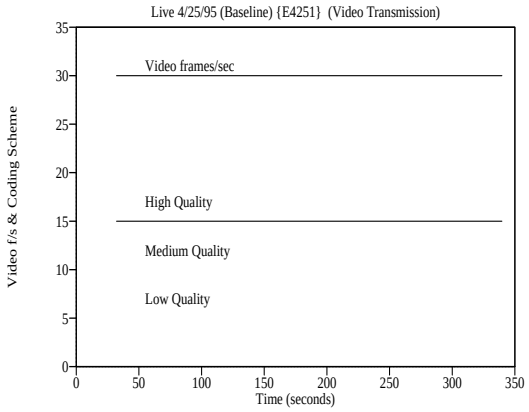
(c) Audio Latency



(d) Video Latency

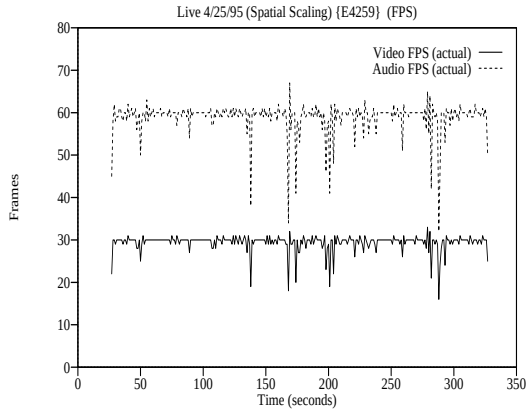


(e) Audio Transmission

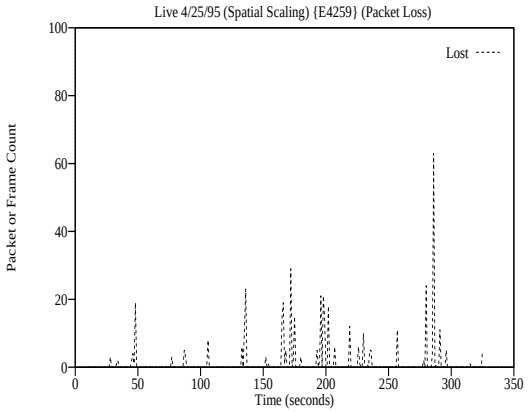


(f) Video Transmission

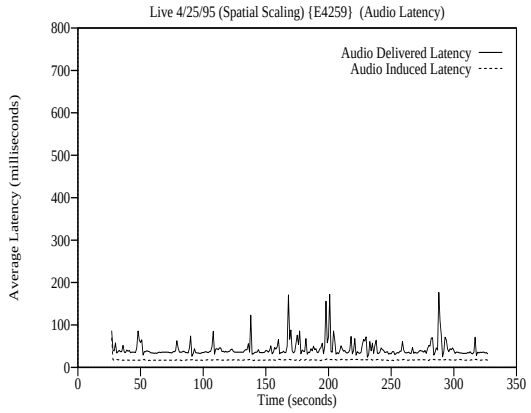
Figure 6-22: Sitterson Network Experiment - April 25, 1995 - Baseline



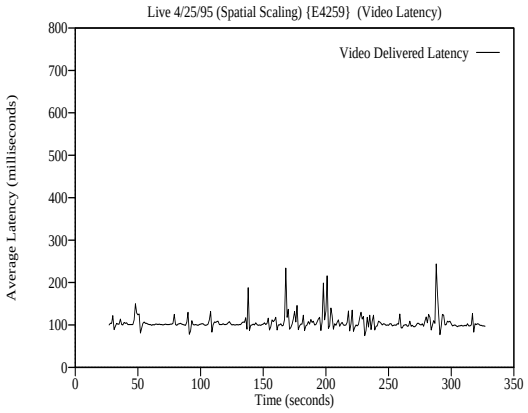
(a) Frames Per Second (FPS)



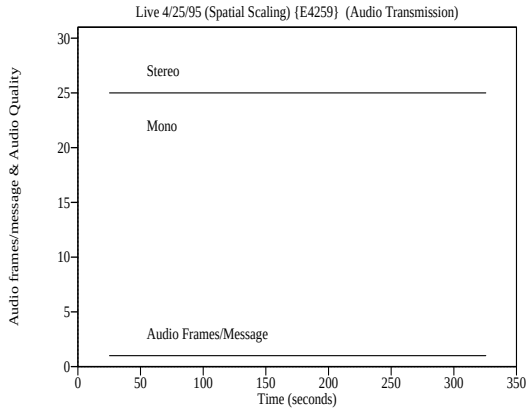
(b) Message Loss



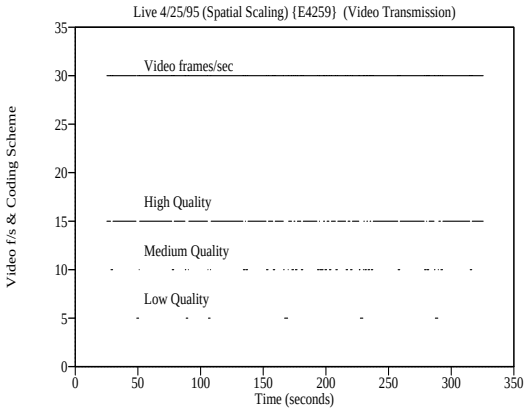
(c) Audio Latency



(d) Video Latency

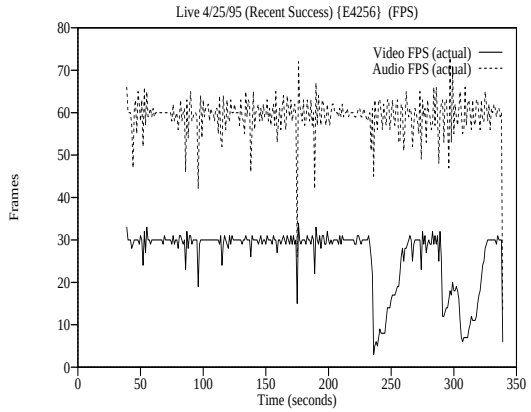


(e) Audio Transmission

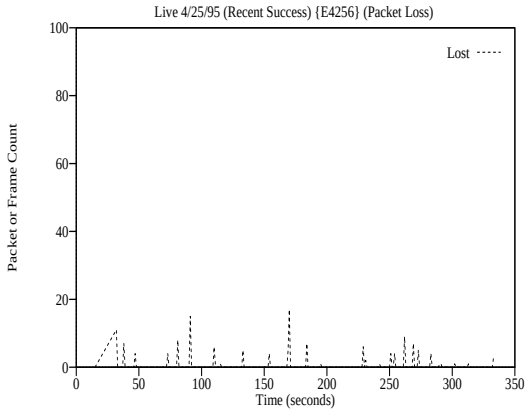


(f) Video Transmission

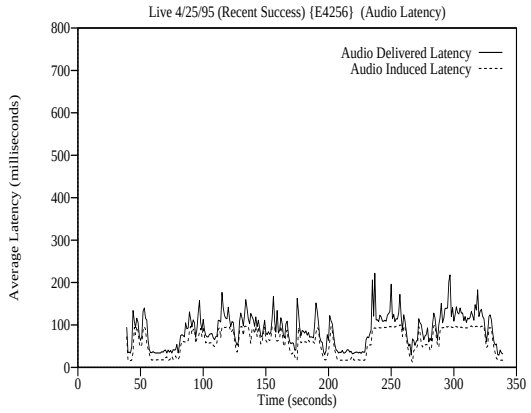
Figure 6-23: Sitterson Network Experiment - April 25, 1995 - Video Scaling Only



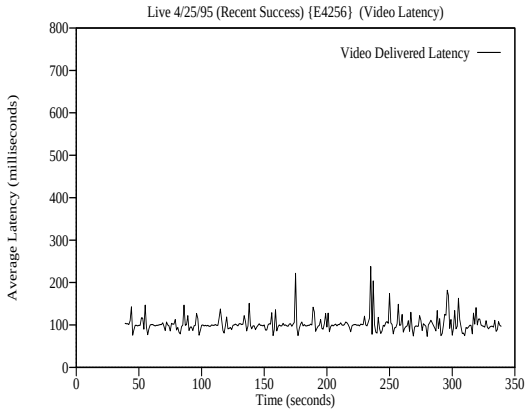
(a) Frames Per Second (FPS)



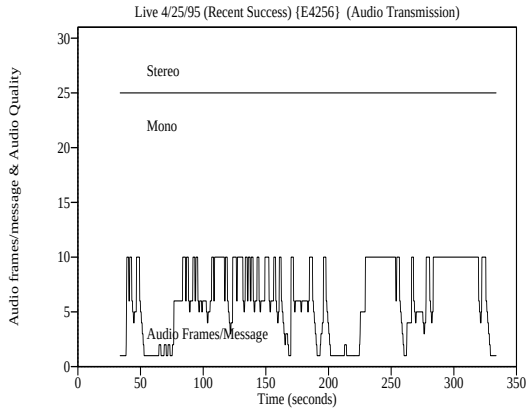
(b) Message Loss



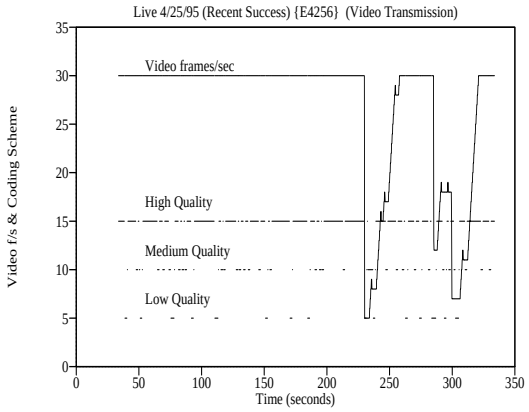
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 6-24: Sitterson Network Experiment - April 25, 1995 - Recent Success

Experiment 4/25	Baseline	VSO	Recent Success
Audio FPS			
Mean	57.08	58.96	59.22
Standard deviation	6.50	3.80	5.14
Minimum	8	32	13
Maximum	65	67	73
Mode/Median	60/59	60/60	60/60
Gaps	849	304	214
Audio Latency: Delivered (Induced)			
Mean	53.43 (17.37)	43.25 (17.64)	88.82 (62.70)
Standard deviation	30.97 (0.65)	19.67 (0.84)	39.01 (29.89)
Minimum	28 (15)	24 (16)	29 (12)
Maximum	203 (19)	177 (20)	222 (100)
Mode/Median	34 (17)/40 (17)	36 (18)/36 (18)	34 (17)/88 (62)
Intervals > 250 ms	0	0	0
Audio Messages			
Frames Sent	18501	18073	18026
Msgs(frames) Lost	828 (828)	284 (284)	52 (167)
Mean Frames Lost	2.68	0.94	0.55
Max Frames Lost	38	43	18
Video FPS			
Mean	28.28	29.43	26.39
Standard deviation	3.63	1.95	7.09
Minimum	4	16	3
Maximum	32	33	34
Mode/Median	30/30	30/30	30/30
Gaps	506	168	1063
Video Latency: Delivered			
Mean	116.58	105.77	102.38
Standard deviation	27.99	18.55	19.68
Minimum	95	75	73
Maximum	279	244	238
Mode/Median	98/104	101/101	99/99
Intervals > 250 ms	1	0	0
Video Messages			
Frames Sent	9250	9036	8057
Frames Lost	494	156	93
Mean Frames Lost	1.60	0.52	0.31
Max Frames Lost	21	20	12

Table 6-8: Summary Statistics for April 25 Experiments

6.4.7. Results for Wednesday, April 26, 1995

Figures 6-25, 6-26, 6-27, and 6-28 show the results for a set of conferences using the Baseline (BL), Video Scaling Only (VSO), Temporal Scaling Only (TSO), and Recent Success (RS) algorithms. The conferences described in this section were run between 11:00 and 11:30 on April 26, 1995. Table 6-9 summaries the results. Figure 6-25 shows that the performance of the BL algorithm under heavy congestion can be quite bad. Part (a) shows very poor delivered audio and video rates. Each stream experiences many gaps (*i.e.*, 4,163 audio gaps and 2,453 video gaps; Table 6-9). Audio quality is poor throughout the conference and often unintelligible. Both audio and video experience high latencies (parts (c) and (d)) and high message loss (part (b)).

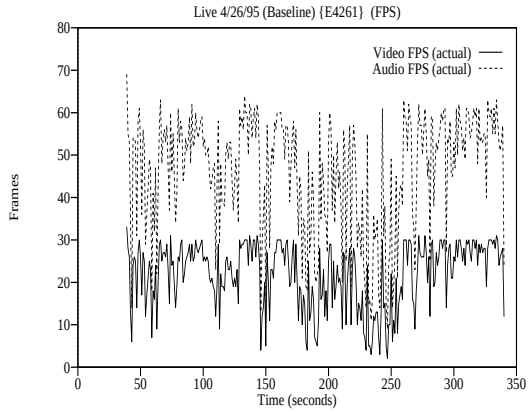
RS delivers a better conference than BL (Figure 6-28). The delivered audio frame rate is good and video frame rate varies with network load (part (a)). Audio and video latencies are low (parts (c) and (d)) and there is low message loss (part (b)). There are 220 audio gaps and the average delivered video frame rate is 27 FPS (Table 6-9). The lower delivered video frame rates are a direct result of the lower transmitted video frame rates during the congested periods. The network congestion begins to subside at approximately 270 seconds into the RS conference. The VSO conference, which we ran immediately after the RS conference, produces a good quality conference (Figure 6-26 (a), (b), (c), and (d)). The VSO algorithm must make only occasional short adjustments to the video quality (part (f)) to deal with the light congestion.

This suite of experiments also includes an experiment using video temporal scaling. Figure 6-27 shows the results using the Temporal Scaling Only (TSO) algorithm. The results are similar to those observed with RS (albeit with slightly lower video frame rates in the case of TSO; see Table 6-9). This is not too surprising since in the Ethernet environment temporal scaling dramatically reduces the packet rate on the LAN. It appears that for the level of network congestion present in these experiments the relatively small reduction in audio packets produced by audio packaging has little impact compared with the reduction of packets due to temporal video scaling. If the level of traffic on the Ethernet increased, audio scaling may become necessary to achieve the required reduction in packet rate.

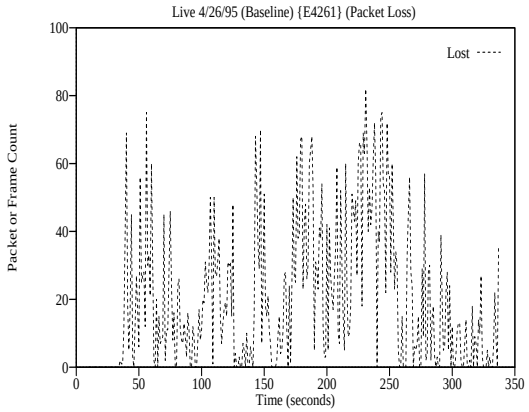
The experiments in this section again demonstrate that a non-adaptive algorithm like BL is inadequate with heavy congestion and can produce extremely poor conference quality. Depending on the level of congestion, TSO may achieve similar results to RS, but RS outperforms TSO under heavier network loads (see Chapter 5). In earlier experiments VSO also achieved similar results to RS. However, since RS is a superset of VSO and TSO, it is unlikely VSO or TSO will ever significantly outperform RS. On the other hand, we have demonstrated that there are situations where RS will significantly outperform VSO and TSO (see Chapter 5).

6.4.8. Results for Friday, April 28, 1995

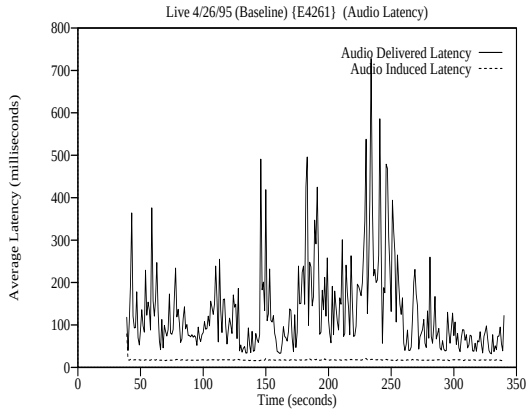
We ran the last set of production network experiments between 10:00 a.m. and 10:30 a.m. on April 28, 1995. Figures 6-29, 6-30, 6-31, and 6-32 show the results of experiments using the BL, VSO, TSO, and RS algorithms, respectively. Table 6-10 summarizes the results. As with the previous production network experiments, RS produces better audio quality and loses fewer messages than either the BL or VSO algorithms. As in the April 26 experiments, TSO gives roughly equivalent performance to RS, which is not surprising for the network topology and traffic load (*i.e.*, the network is very lightly loaded during the TSO conference; note in Figure 6-31 that the algorithm only rarely uses less than full rate video). The variation in the delivered audio frames for RS and TSO (Figures 6-32 (a) and 6-31 (a), respectively) results primarily from RS packaging multiple audio frames into a single message. The fidelity of the played audio is roughly equivalent. TSO delivers lower audio latency than RS because of the induced latency associated with audio packaging.



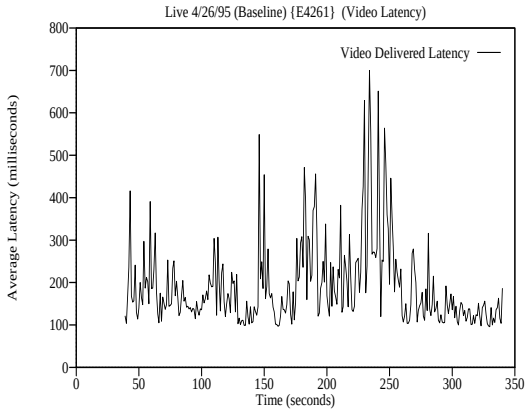
(a) Frames Per Second (FPS)



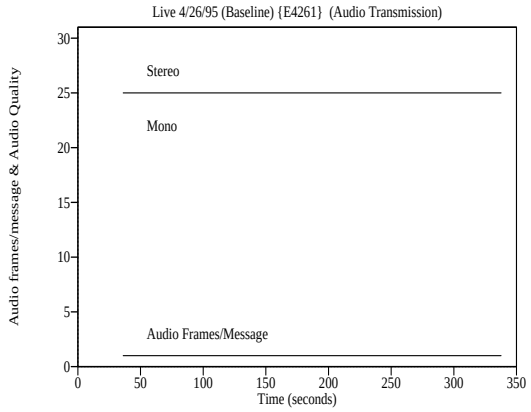
(b) Message Loss



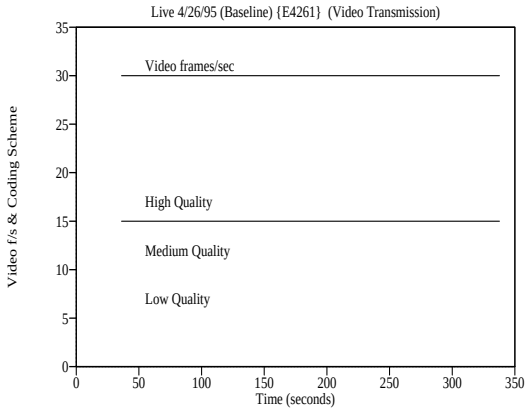
(c) Audio Latency



(d) Video Latency

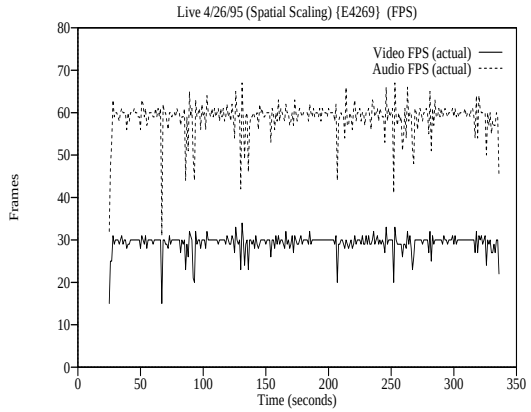


(e) Audio Transmission

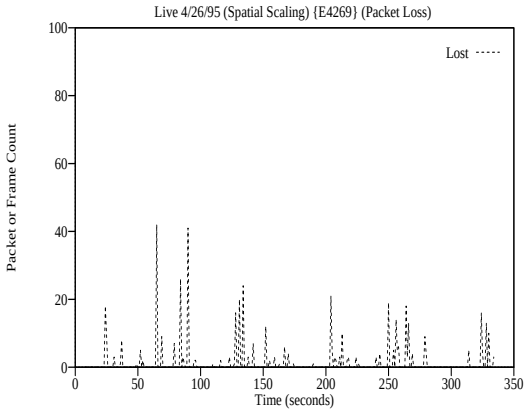


(f) Video Transmission

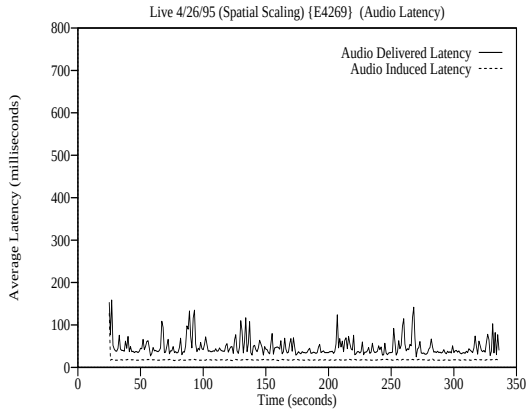
Figure 6-25: Sitterson Network Experiment - April 26, 1995 - Baseline



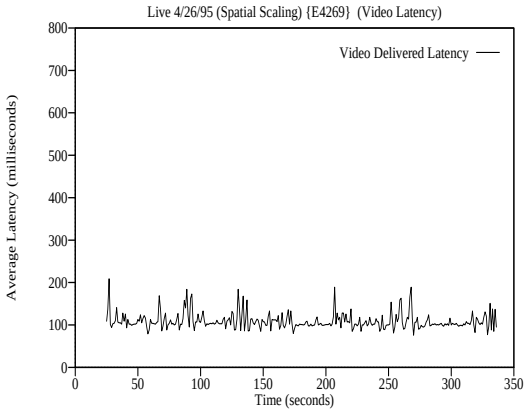
(a) Frames Per Second (FPS)



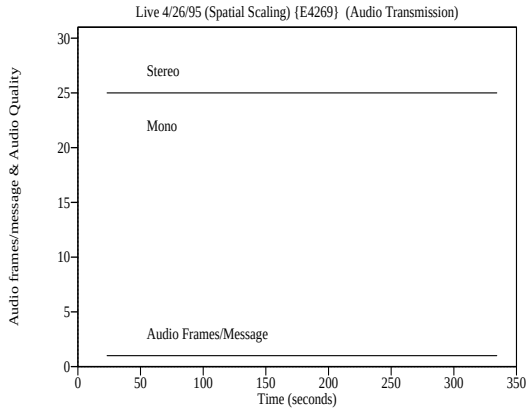
(b) Message Loss



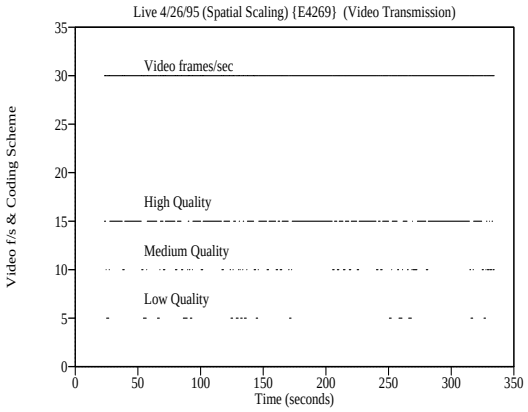
(c) Audio Latency



(d) Video Latency

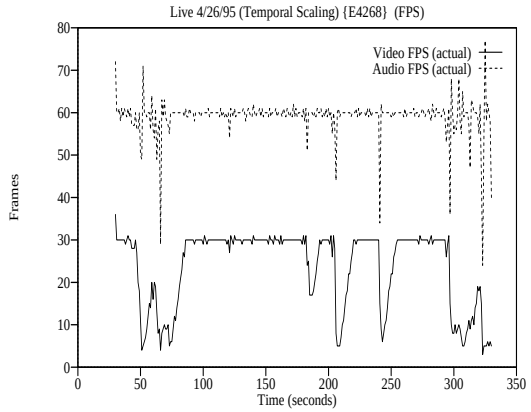


(e) Audio Transmission

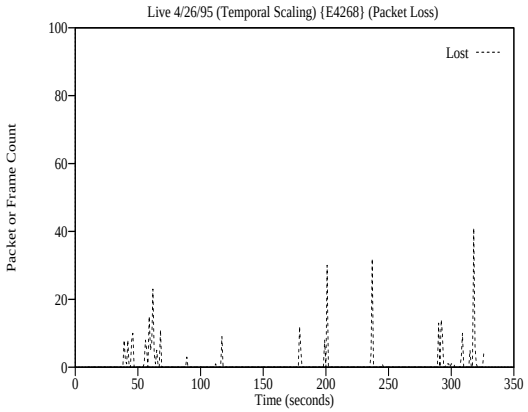


(f) Video Transmission

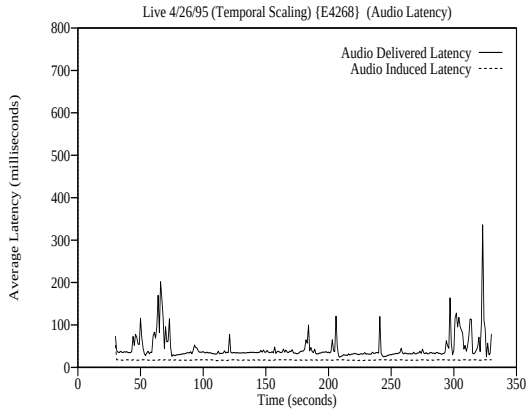
Figure 6-26: Sitterson Network Experiment - April 26, 1995 - Video Scaling Only



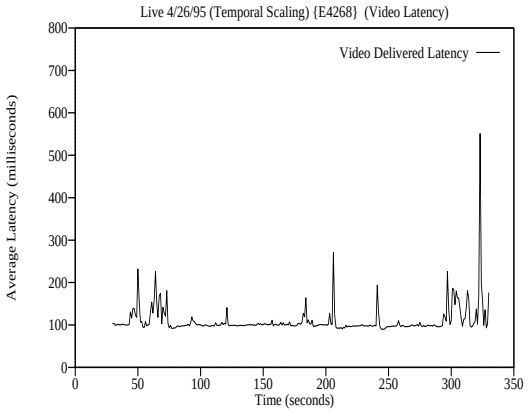
(a) Frames Per Second (FPS)



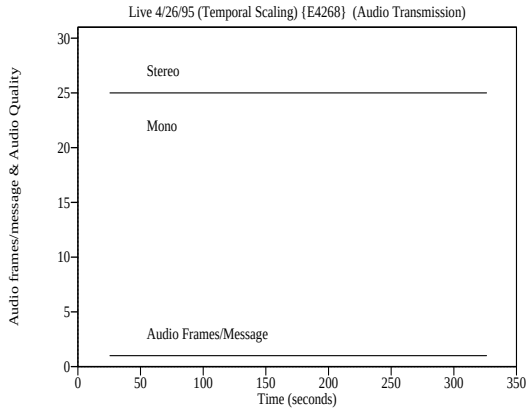
(b) Message Loss



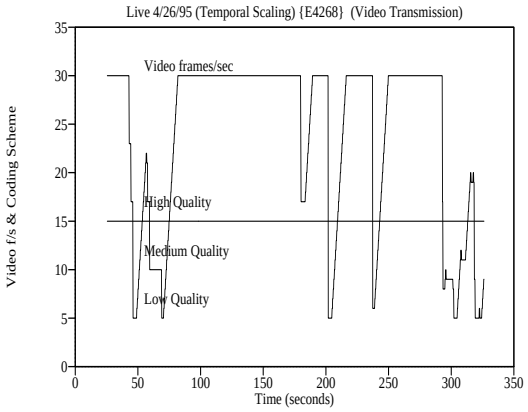
(c) Audio Latency



(d) Video Latency

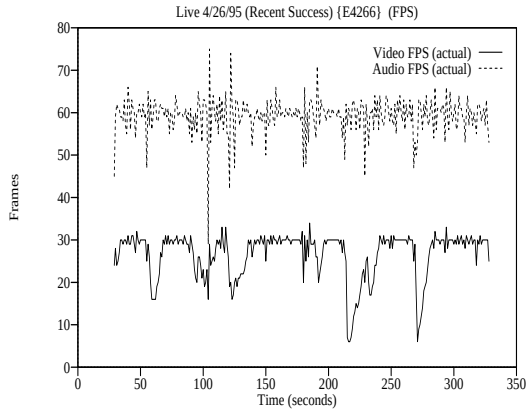


(e) Audio Transmission

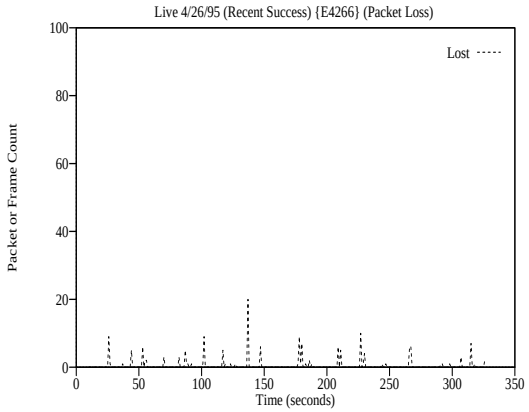


(f) Video Transmission

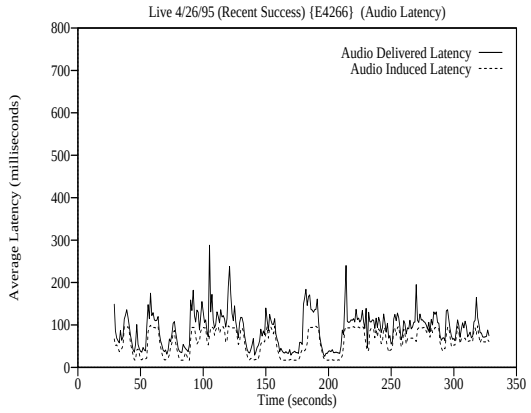
Figure 6-27: Sitterson Network Experiment - April 26, 1995 - Temporal Scaling Only



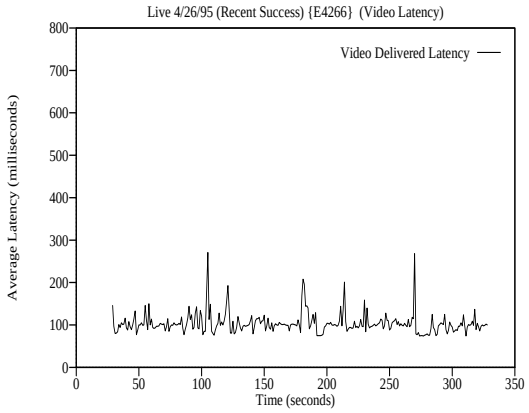
(a) Frames Per Second (FPS)



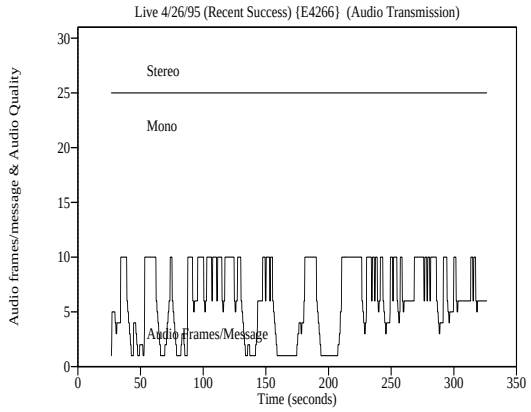
(b) Message Loss



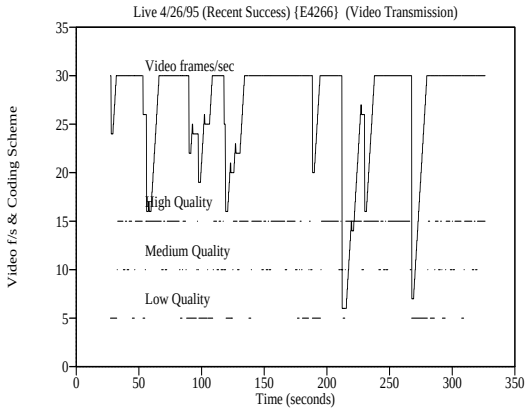
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission

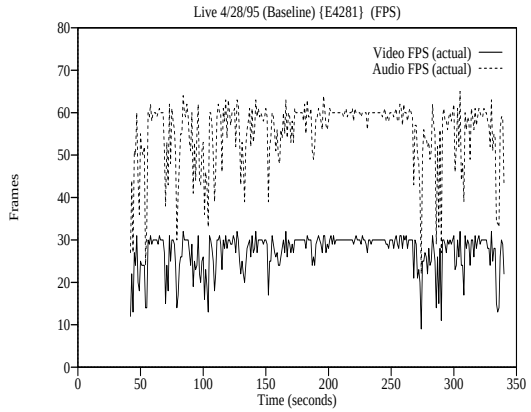


(f) Video Transmission

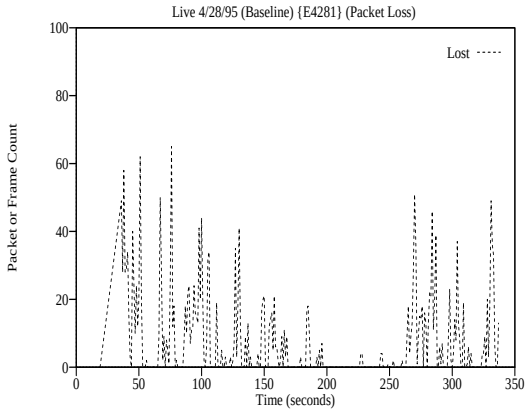
Figure 6-28: Sitterson Network Experiment - April 26, 1995 - Recent Success

Experiment 4/26	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	46.09	58.89	59.06	59.30
Standard deviation	13.49	3.87	4.45	4.24
Minimum	9	31	24	29
Maximum	64	67	77	95
Mode/Median	53/50	60/60	60/60	60/60
Gaps	4163	338	276	220
Audio Latency: Delivered (Induced)				
Mean	131.91 (17.35)	47.85 (17.74)	44.98 (17.44)	90.99 (61.91)
Standard deviation	103.40 (0.85)	20.78 (0.75)	30.57 (0.56)	40.78 (28.89)
Minimum	32 (16)	25 (16)	24 (6)	26 (17)
Maximum	728 (21)	159 (20)	336 (18)	288 (99)
Mode/Median	40 (17)/95 (17)	35 (18)/40 (18)	35 (17)/35 (17)	112 (94)/93 (63)
Intervals > 250 ms	31	0	1	1
Audio Messages				
Frames Sent	18143	18718	18052	17993
Msgs(frames) Lost	4152 (4152)	312 (312)	244 (244)	52 (187)
Mean Frames Lost	13.70	1.00	0.81	0.62
Max Frames Lost	54	27	34	18
Video FPS				
Mean	21.82	29.34	23.72	26.66
Standard deviation	7.59	2.03	9.08	5.48
Minimum	2	15	3	6
Maximum	31	34	31	34
Mode/Median	30/24	30/30	30/30	30/29
Gaps	2453	203	1874	1001
Video Latency: Delivered				
Mean	191.05	108.99	110.39	103.65
Standard deviation	99.09	19.11	35.86	24.40
Minimum	96	76	90	74
Maximum	700	209	551	271
Mode/Median	122/156	102/104	99/100	98/99
Intervals > 250 ms	54	0	2	2
Video Messages				
Frames Sent	9071	9360	7258	8115
Frames Lost	2446	188	97	106
Mean Frames Lost	8.07	0.60	0.32	0.35
Max Frames Lost	28	17	11	7

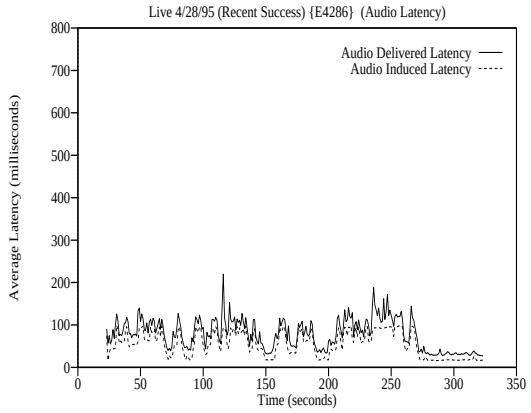
Table 6-9: Summary Statistics for April 26 Experiments



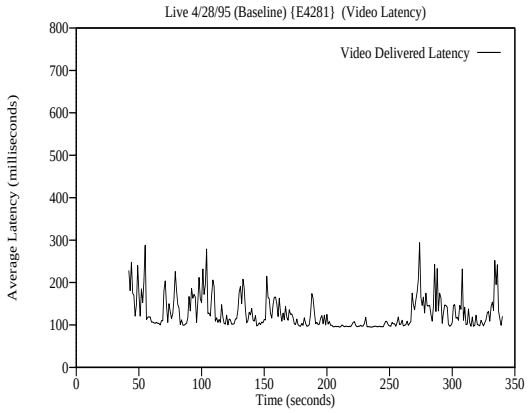
(a) Frames Per Second (FPS)



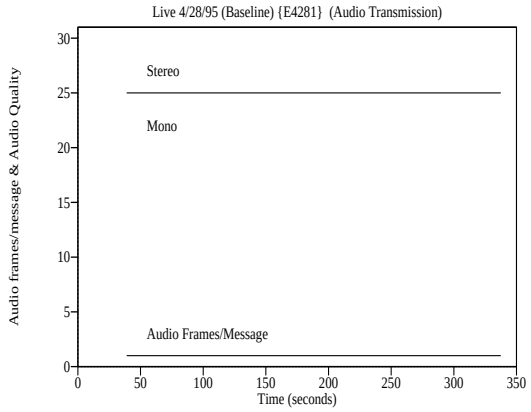
(b) Message Loss



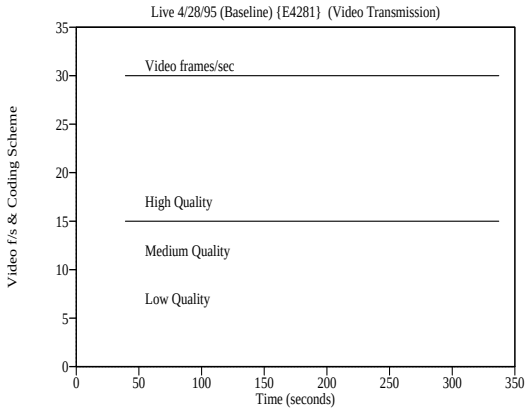
(c) Audio Latency



(d) Video Latency

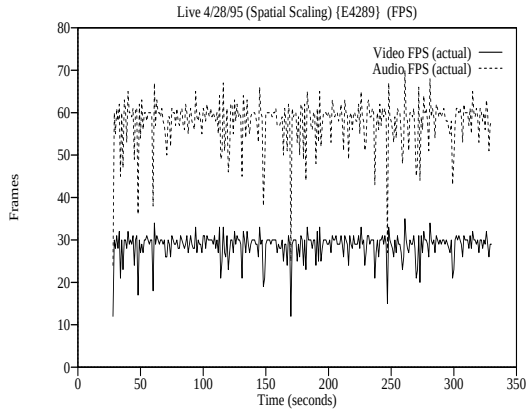


(e) Audio Transmission

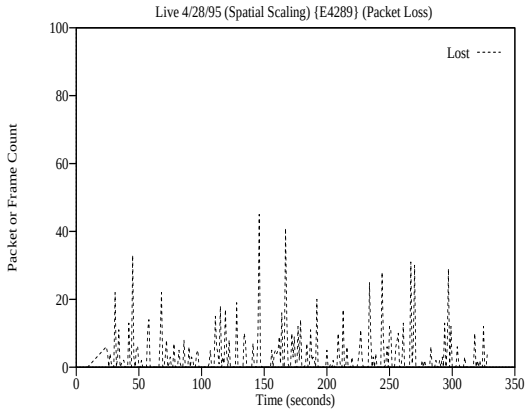


(f) Video Transmission

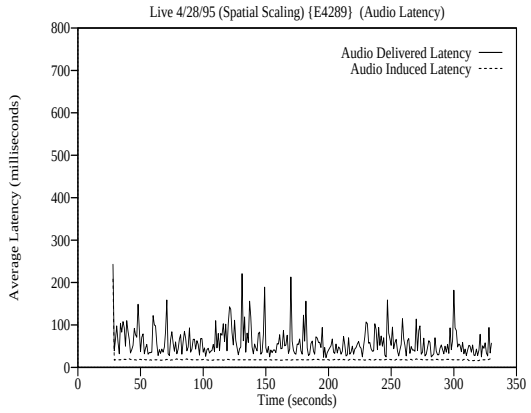
Figure 6-29: Sitterson Network Experiment - April 28, 1995 - Baseline



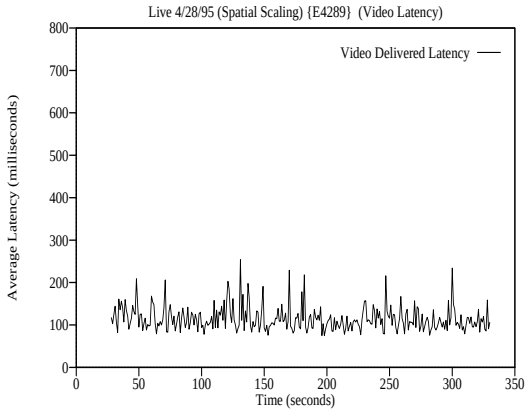
(a) Frames Per Second (FPS)



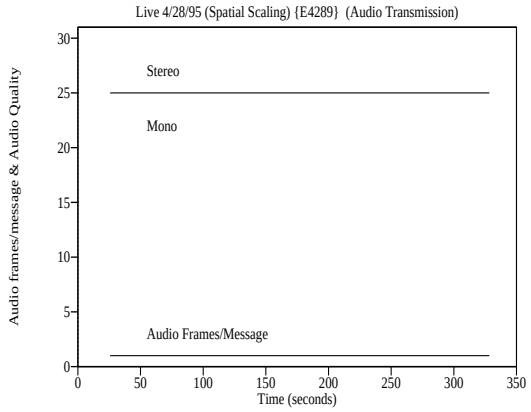
(b) Message Loss



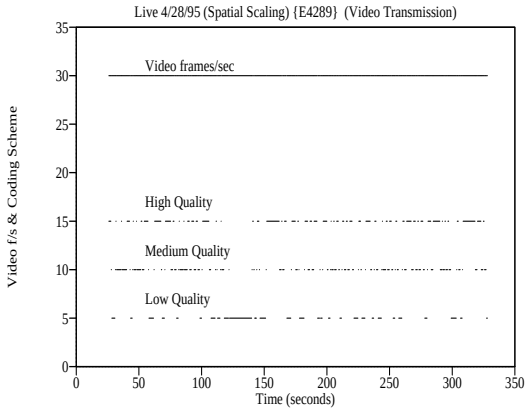
(c) Audio Latency



(d) Video Latency

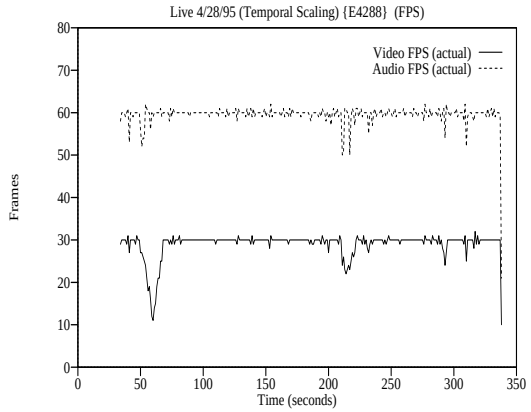


(e) Audio Transmission

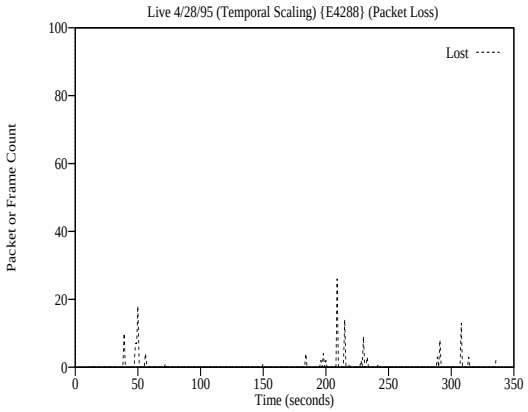


(f) Video Transmission

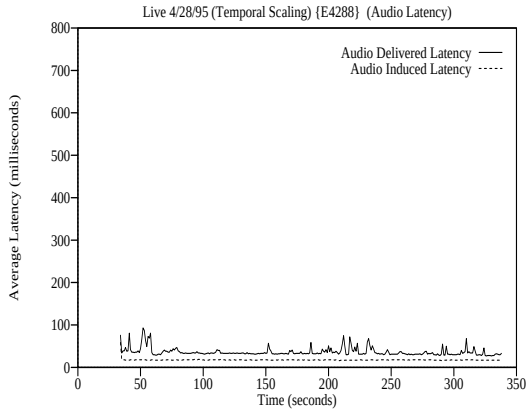
Figure 6-30: Sitterson Network Experiment - April 28, 1995 - Video Scaling Only



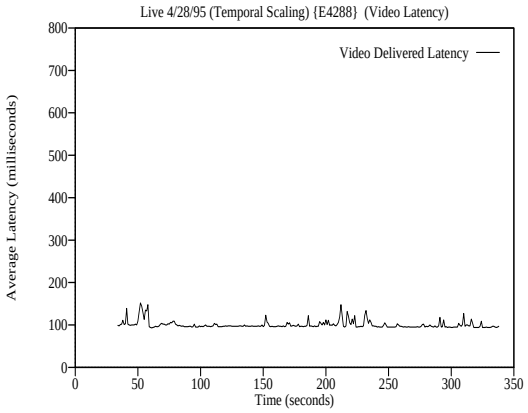
(a) Frames Per Second (FPS)



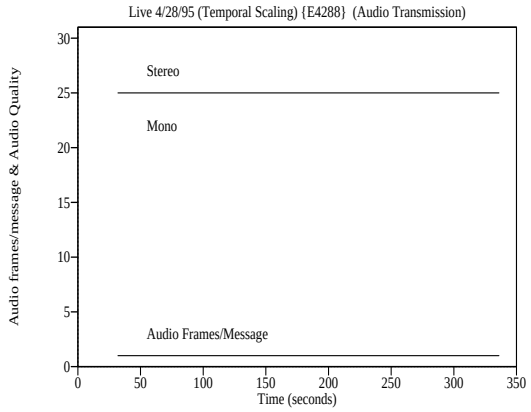
(b) Message Loss



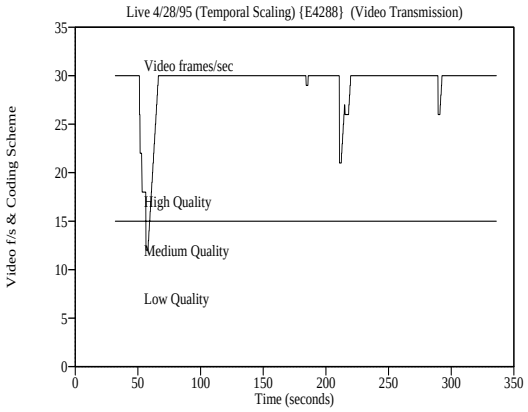
(c) Audio Latency



(d) Video Latency

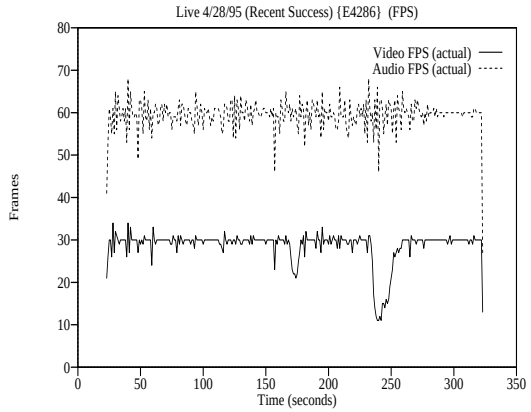


(e) Audio Transmission

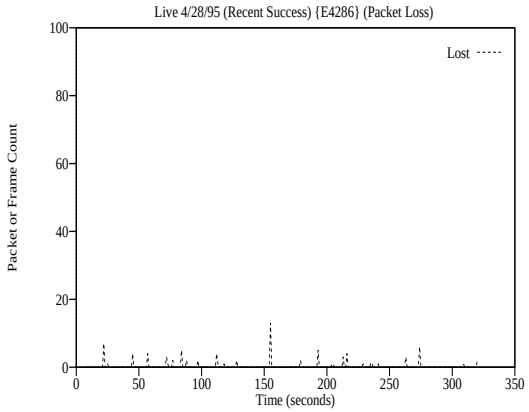


(f) Video Transmission

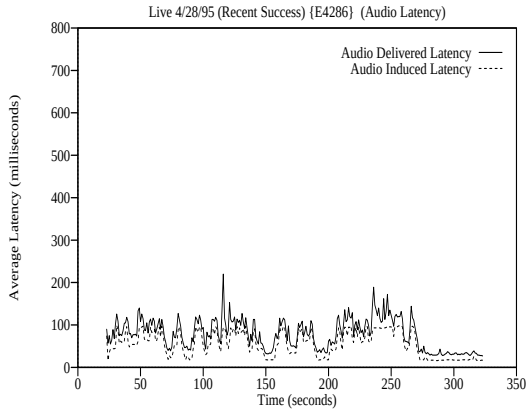
Figure 6-31: Sitterson Network Experiment - April 28, 1995 - Temporal Scaling Only



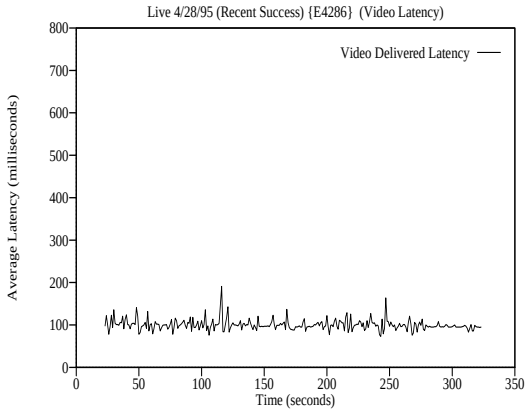
(a) Frames Per Second (FPS)



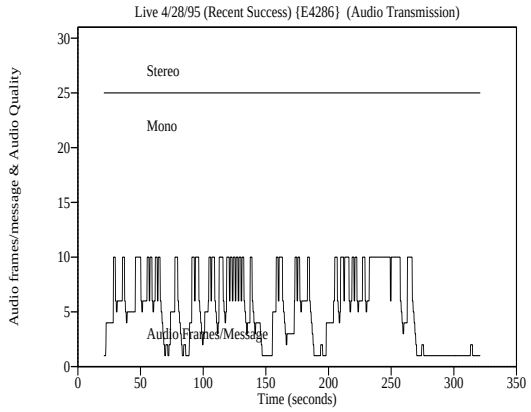
(b) Message Loss



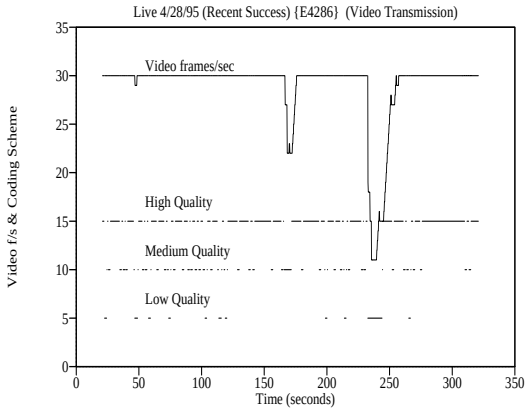
(c) Audio Latency



(d) Video Latency



(e) Audio Transmission



(f) Video Transmission

Figure 6-32: Sitterson Network Experiment - April 28, 1995 - Recent Success

Experiment 4/28	Baseline	VSO	TSO	Recent Success
Audio FPS				
Mean	55.36	57.79	59.50	59.51
Standard deviation	7.75	5.51	2.72	3.46
Minimum	22	25	21	27
Maximum	65	70	62	68
Mode/Median	60/59	60/59	60/60	60/60
Gaps	1450	674	113	123
Audio Latency: Delivered (Induced)				
Mean	66.22 (17.24)	58.67 (18.09)	36.26 (17.30)	78.93 (55.70)
Standard deviation	41.16 (0.65)	31.66 (0.76)	9.79 (0.58)	35.76 (29.45)
Minimum	31 (15)	23 (16)	27 (16)	27 (16)
Maximum	250 (19)	221 (20)	93 (19)	220 (99)
Mode/Median	33 (17)/49 (17)	31 (18)/50 (18)	33 (18)/33 (18)	31 (18)/77 (54)
Intervals > 250 ms	0	0	0	0
Audio Messages				
Frames Sent	17996	18166	18282	18032
Msgs(frames) Lost	1414 (1414)	646 (646)	92 (92)	36 (95)
Mean Frames Lost	4.70	2.13	0.30	0.32
Max Frames Lost	42	29	17	12
Video FPS				
Mean	27.31	28.74	29.01	28.70
Standard deviation	4.38	2.92	3.02	3.82
Minimum	9	12	10	11
Maximum	32	35	32	34
Mode/Median	30/29	30/29	30/30	30/30
Gaps	844	387	284	375
Video Latency: Delivered				
Mean	128.44	114.58	100.53	100.28
Standard deviation	38.10	28.93	9.44	12.95
Minimum	95	75	93	73
Maximum	294	254	152	191
Mode/Median	97/113	99/107	97/97	95/99
Intervals > 250 ms	4	1	0	0
Video Messages				
Frames Sent	8999	9083	8925	8701
Frames Lost	826	371	56	49
Mean Frames Lost	2.74	1.22	0.18	0.16
Max Frames Lost	23	16	9	7

Table 6-10: Summary Statistics for April 28 Experiments

6.5. Overall Conclusions from Sitterson Network Tests

This chapter has discussed a series of experiments on a production network using four distinct transmission control strategies. These experiments were on a single production network and are not intended to represent all possible network configurations and traffic patterns. Instead, they show that the conditions discussed and demonstrated in the controlled network experiments do occur on well-managed, production networks. Table 6-11 gives a qualitative summary of the results of these experiments.

	Qualitative Assessment of Results
April 18, 1995	Light network congestion. RS produces slightly better results than BL and VSO, but all are good. RS and VSO impose no performance penalty under light load.
April 19, 1995	Heavy congestion during BL and RS conferences. BL gives poor performance. RS significantly outperforms BL. Congestion is lighter during the VSO conference, but RS still gives better audio performance than VSO.
April 20, 1995 (set 1)	Similar results to April 18 conferences.
April 20, 1995 (set 2)	Very heavy congestion during all of the BL and RS conferences. Very heavy congestion during first half of VSO conference. Poor performance from BL. RS gives good audio quality, but only 12 FPS for video. During congested period, VSO audio performance is much worse than with RS. VSO has much better video frame rates than RS, but we prefer the RS conference because of audio performance.
April 21, 1995	Heavy congestion during the VSO conference. Comparing results with April 20 (set 2), we conclude RS performs better under heavy congestion than BL and VSO.
April 24, 1995	Similar results to April 18 conferences.
April 25, 1995	Mainly light congestion with short periods of moderate congestion. RS aggressively responds to congestion and may make drastic short term reductions in the video frame rate during brief periods of congestion. We prefer aggressive reaction to limit congestion.
April 26, 1995	Heavy congestion during the BL conference and very poor performance. Light congestion during VSO, TSO, and RS conferences and all deliver roughly equivalent performance.
April 28, 1995	Light to moderate congestion during BL, VSO, and RS conferences. RS delivers the best results. Congestion very light during TSO (essentially no adaptations required) and performance is comparable to RS.

Table 6-11: Summary of Production Network Experiments

In the production network experiments, the Recent Success algorithm consistently delivered conferences with fewer audio gaps than other algorithms. The differences

between the number of gaps increased as the level of congestion increased. The conferences using RS always had the lowest overall message loss and the lowest message loss within any particular measurement interval. The conferences using RS also always had the lowest video latency and usually had competitive video frame rates. In the case when the video frame rates were not competitive (the second set of experiments on April 20), RS delivered a significantly better audio stream than either the BL or VSO algorithm. Given the relative importance of audio over video in a video conference, we think the conference with improved audio performance is preferable.

The experiments of this chapter reinforce those on the controlled network. The Recent Success algorithm performs as well as any of the other algorithms considered and significantly outperforms all other algorithms under specific conditions and under heavy network loads. This is the case in both the controlled network experiments and in the production network experiments. Under light loads, RS imposes no performance penalties. RS performs as well as or better than the video scaling algorithms on all networks, regardless of the size of the network MTU or whether or not messages are fragmented during transmission. These results are not surprising since RS uses both message rate and bit rate adaptations to control transmission and these adaptations are a superset of existing scaling techniques. There are clearly traffic levels for all production networks that prohibit any best effort transmission strategy from delivering a quality conference. However, the experiments in this chapter show that production networks can usually support acceptable quality conferences even when the network is congested provided we use RS or a similar algorithm based on the transmission control framework to control transmission of the audio and video streams.

Chapter VII
Summary and Conclusions

The computer is an increasingly important communications tool. Unfortunately, most forms of computer-based communication are not conversational. We would like to extend the capabilities of desktop video conferencing systems to support the same style of conversational interaction people use when talking face-to-face. In particular, we want to support high fidelity, low latency video conferences over current networks such as token rings, Ethernets, and T1 lines.

Providing a high quality video conference requires careful management of the resources at the conference endpoints and a good transmission control policy for transmitting the media streams from source to destination. In this dissertation, we have focused on how one can avoid or ameliorate the problems associated with transmitting a video conference over "best-effort" networks. Best-effort networks do not provide any guarantees on message delivery and congestion can severely degrade the quality of a video conference. In contrast, networks supporting resource reservation avoid congestion by reserving network resources on behalf of individual users. Reservation-based systems can consistently provide good quality conferences, but providing guaranteed network performance is complicated and expensive and, as a result, is not available on most networks. Furthermore, on some multi-access networks, it is impossible to provide guaranteed service. Best-effort systems are unlikely to be completely replaced by reservation-based systems. A best-effort system will be used even if the overall quality sometimes suffers in comparison to the reservation-based system because it is less costly. The current interest in Internet-based telephones [72] is a good example of this phenomenon. The conventional telephone system has high quality, but appears expensive compared with the cost of using the Internet (ignoring the initial costs of buying a computer and getting an Internet service provider). As a result, a substantial number of people are using the Internet-based phone alternatives, even though the quality is significantly lower than that provided by the conventional telephone system. Ideally, best-effort systems would not only be inexpensive, but also preserve the quality of the conference

when competition for network resources adversely affects the transmission of the media streams.

Our goal in this dissertation is to determine how to transmit the audio and video data streams of a conference over best-effort networks so that the resulting fidelity and the latency of the streams are sufficient to provide adequate quality. We have made four contributions towards this goal:

1. We created a transmission control framework for continuous media that describes the capabilities of a conferencing system and relates these capabilities to human perception and network congestion.

2. We described the effects of network congestion on a video conference, articulated the two types of network constraints that cause congestion, and demonstrated how to exploit the characteristics of the audio and video streams to address these constraints.

3. We developed a transmission control algorithm, called *Recent Success*, that is based on our transmission control framework. We implemented the algorithm in C and integrated that implementation with an experimental video conferencing system.

4. We demonstrated that Recent Success can deliver high fidelity, low latency conferences on congested networks.

The rest of this chapter discusses each of these contributions in more detail and concludes with a statement of some possibilities for future research.

7.1. A Transmission Control Framework

We developed a framework for transmission control of audio and video streams across best-effort networks. This transmission control framework describes how changes to the message and the bit rates of the audio and video streams affect the delivered quality of the conference. The framework also shows how we can manipulate these rates to adapt to both static and dynamic network constraints.

The transmission control framework describes the capabilities of a video conferencing system using a simple abstraction, a set of operating points. The operating points

describe all combinations of message rate and bit rate available for each media stream. The conferencing application is free to choose any one of the operating points as the current operating point for a particular media stream. However, the choice of operating point may affect the delivered quality of the media stream. The fundamental problems of transmission control are estimating the level of congestion, deciding when to change the operating point for a media stream, and selecting a new operating point.

The abstract representation of the conferencing system as a set of operating points provides all the information required for cooperative, adaptive control of the media streams by the transmission control policy and the *media generation subsystem* (*i.e.*, the digitization, compression and packaging stages of the conferencing pipeline; see Figure 7-1).

The transmission control policy communicates with the media generation subsystem by selecting and setting the current operating point which indirectly sets the parameters for the digitization, compression, and packaging stages. The transmission control policy is responsible for selecting the particular operating point to be used over some interval and the media generation subsystem is responsible for producing the stream. The media generation subsystem does not explicitly deal with network congestion, but simply produces media streams based on the operating point set by the transmission control policy. The transmission control policy is not concerned with the specifics of how the media stream associated with a particular operating point is produced. The control policy simply selects an operating point with a message and bit rate that can be effectively transmitted in the current network environment. This separation of concerns and limited interface allows us to build a very general transmission control policy that can be used in a variety of video conferencing systems.

Knowing the operating points available for a particular media stream and being able to select the current operating point provides only the mechanism for building a transmission control algorithm. The transmission control framework also relates human perception and network constraints to the conference operating points and the delivered conference quality. Knowing these relationships helps the transmission control policy select operating points so that the delivered media streams have acceptable fidelity and latency.

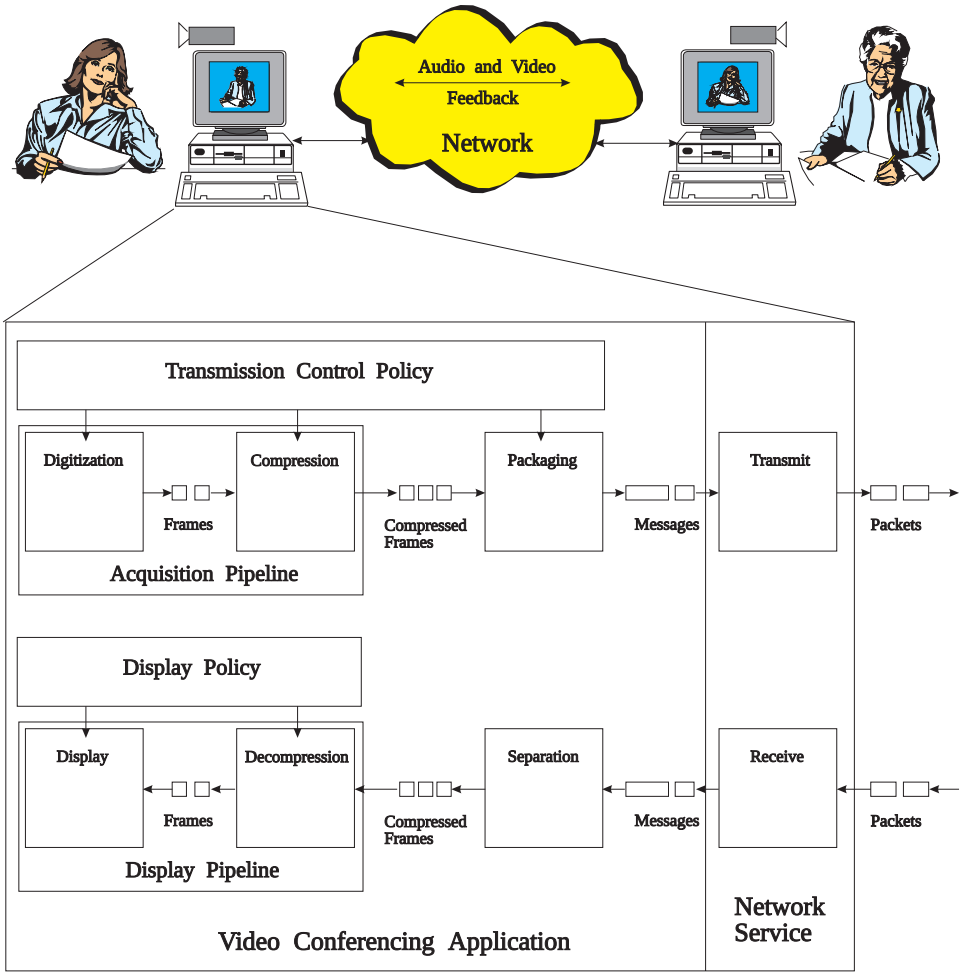


Figure 7-1: Video Conferencing Application Architecture

For example, human perception may constrain the choice of operating points because of either latency or fidelity concerns. The transmission control framework describes when and which operating points are excluded as a result of perceptual constraints. Some potential operating points may be eliminated because of static perceptual constraints. For example, if 250 milliseconds is the maximum acceptable stream latency, then all operating points with message rates below 4 messages per second do not provide acceptable latency and cannot be used. Message rates below 4 messages per second causes the latency induced by buffering to exceed our latency guideline. This is a static constraint because the constraint does not change over time. There may also be dynamic perceptual constraints. These constraints vary over time because of changes in the network conditions. If we assume a maximum acceptable latency of

250 *ms*, an operating point with induced latency of 150 *ms* is perceptually feasible if the transmission latency is 50 *ms* (since 150 + 50 < 250), but is not feasible if the transmission latency is 120 *ms* (since 150 + 120 > 250).

7.2. Effects of Network Congestion

The transmission control framework explains the effects of network congestion using a simple queueing model. We demonstrated how capacity and access constraints cause congestion and showed the relationship of bit and message rates to these constraints. Capacity constraints limit the sustainable bit rates in the network and are caused by either (1) limited network bandwidth on a transmission link, or (2) internal data movement time at a router being the dominant component of the service demand. Capacity constraints are affected only by the stream bit rate and are not affected by the message rate or resulting packet rate. Access constraints limit the sustainable packet rates in the network and are caused by (1) medium access times when transmitting across shared-medium networks, or (2) packet processing time at a router being the dominant component of the service demand. Access constraints are affected by the message rate or resulting packet rates on a hop and not by the bit rate. Capacity and access constraints may occur in the network either separately or in combination. We further categorized capacity and access constraints as either structural or congestion constraints. Structural constraints are static and result from physical limitations in the network. Congestion constraints are dynamic and result from transient traffic conditions.

Capacity and access constraints limit network performance in fundamentally different ways. Addressing the two different constraints requires different techniques. The transmission control policy must address capacity constraints by reducing the stream bit rate. Media scaling is particularly well suited to addressing capacity constraints. On the other hand, the transmission control policy must address access constraints by reducing the stream message rate. Media packaging is an excellent strategy for addressing access constraints, particularly with networks where the MTUs are large in relation to the media frames. The concepts of capacity and access constraints are well known. Router manufacturers often advertise supported bit and packet rates separately. However, most conferencing systems do not exploit the differences in these distinct types of constraints. To be successful, a control scheme must address both types of constraints.

It is particularly important to address access constraints since it has been our experience and the experiences of other network managers [115, 117] that dynamic access constraints are far more common on existing LANs than dynamic capacity constraints. Unfortunately, most existing video conferencing systems either do not adapt at all or rely solely on video bit rate scaling to address network congestion. While video scaling is a good response to capacity constraints, it is a poor response to access constraints. With video scaling, the effective packet rate is changed only as a secondary effect of the reduction in stream bit rate. Best-effort schemes must use both scaling and packaging techniques to be successful over a wide range of network topologies, technologies, and load levels. Transmission control schemes based on our transmission control framework are fundamentally better able to adapt to network congestion than bit rate scaling techniques because algorithms based on the framework incorporate two-dimensional control of the media streams whereas bit rate scaling algorithms adapt along only a single dimension.

Message fragmentation may exacerbate access constraints, because fragmentation can greatly increase the number of packets generated on a hop. We demonstrated how we can calculate the realization of a message rate in terms of packet rate for any network hop given the network fragmentation algorithm. We can incorporate the realization into a three-dimensional representation of the transmission control framework for any hop. We can only predict the success of packaging changes on a particular hop from this three-dimensional representation since the effectiveness of packaging media is directly related to the number of packets generated on the hop. Fortunately, the realized packet rates are related to the source message rate and we demonstrated that manipulation of the message rate can address access constraints even when messages are fragmented.

7.3. A Transmission Control Algorithm

The goal of transmission control for video conferencing is to select a feasible operating point, an operating point that provides adequate conference quality and is sustainable under the current network conditions. The transmission control framework mathematically describes the effects of capacity and access constraints on the set of feasible operating points. With perfect knowledge of the network topology and the current network traffic, we could compute the current set of feasible operating points. Unfortunately, end-to-end transmission control schemes cannot determine the exact

state of the network path and thus cannot exactly compute the set of feasible operating points. Furthermore, with end-to-end transmission control techniques, it is impossible to distinguish between the effects of capacity or access constraints or between structural or congestion constraints. Any end-to-end transmission control algorithm can only estimate the set of feasible operating points using a set of heuristics based on information available at the conference endpoints.

We have developed a heuristic transmission control algorithm, called *Recent Success*. The Recent Success algorithm is essentially a search algorithm across the set of operating points. The search is directed by feedback from the conference partner and based on the success of changes to the stream in the recent past. Recent Success uses a set of heuristics to narrow the search space based on the perceived current network conditions. This algorithm is based on the transmission control framework. As such, the algorithm can implement a variety of transmission control strategies ranging from pure scaling to pure packaging algorithms depending on the available set of operating points. Since the algorithm adapts based only on the defined set of operating points and feedback from the conference destination, we can use the algorithm in any video conferencing system. Even though the transmission control framework is conceptually complicated, the resulting algorithm is easy to implement and inexpensive to execute. We have described one implementation of the algorithm and used that implementation in a number of experiments. These experiments show that despite the simplicity of the algorithm, the algorithm is very successful at adapting to diverse network conditions regardless of the type of network congestion or the topology of the network.

7.4. Experimental Results

We have demonstrated the ability of Recent Success to deliver high quality conferences on both experimental and production networks. We have compared the performance of the Baseline (BL), Video Scaling Only (VSO), Temporal Scaling Only (TSO), and Recent Success (RS) algorithms. BL is non-adaptive; it always uses the same video and same audio operating points. VSO uses spatial scaling to scale the video bit rate by changing the coding scheme. TSO uses temporal scaling to scale the video bit and message rate by transmitting fewer frames. RS uses both spatial and temporal scaling and also adaptively packages media frames. We compared these algorithms on access constrained networks, capacity constrained networks, and

networks with a combination of access and capacity constraints. There are several interesting results from these experiments:

- RS always produced conferences with better overall fidelity, latency, and message delivery than the other algorithms.
- RS consistently delivered conferences with fewer audio gaps.
- RS always had the lowest overall message loss and the lowest message loss within any particular measurement interval.
- The conferences using RS always had the lowest video latency and usually had competitive video frame rates.
- Under light loads, RS imposes no performance penalties.
- RS performed as well as or better than any of the other algorithms on all the networks evaluated, regardless of the size of the network MTU or whether or not messages were fragmented during transmission.
- RS performed as well as or better than any of the other algorithms under all types of constraints.
- RS outperformed all other algorithms when the network was heavily congested.
- The degree to which RS outperformed the other algorithms increased as congestion increased.

Perhaps the most surprising result of the experimental and production network experiments was the performance achieved with RS when messages were fragmented during transmission. We demonstrated that fragmentation has little effect on capacity constraints, but can exacerbate access constraints. However, fragmentation does not *always* exacerbate access constraints. If messages are fragmented after they have already crossed the access constrained hop, fragmentation does not increase the number of packets on the constrained hop and so does not affect the access constraint. However, we demonstrated that if messages are fragmented before they cross an access constrained hop, the increase in packets can dramatically lower the quality of the conference. RS cannot directly affect (or even be aware of) the effects of fragmentation in the network. RS can only adjust the packaging of frames into

messages at the conference source. Nevertheless, we demonstrated that even if messages are fragmented before reaching an access constrained hop, packaging at the conference source is still more effective than not performing packaging because the decreasing message rates lead to efficient packing of media frames into packets when fragmentation does occur.

Continuous media applications must control what portions of the media streams are delivered when the network is overloaded. Without this control, all decisions about which portions of the media stream to drop are left to the network. Few network elements have enough knowledge about the media streams to make good decisions about the portions of the stream to throw away. Allowing the network to decide which portions to drop usually results in low quality conferences. This was evident in the BL experiments. In our scheme, the transmission control policy decides what portion of the full media stream to deliver. When network conditions prevent the entire media stream from being delivered, the transmission control policy adjusts the transmission characteristics of the stream to better match the current capabilities of the network.

End-to-end transmission control is not a panacea. For any given network, there are clearly traffic levels that prohibit any best-effort transmission strategy from delivering a quality conference. However, we claim that production networks with unreserved resources can usually support video conferences with adequate fidelity and sufficiently low latency, even under heavily congested conditions, provided Recent Success or a similar algorithm based on our transmission control framework is used to control transmission of the audio and video streams.

7.5. Observations, Recommendations, and Heuristics

Our experiences with transmission control of audio and video have led us to develop a set of observations or rules of thumb that are useful when building a video conference application or a network to support video conferencing. First, when attempting to adapt to a congested network, do not scale media streams until after attempting to ameliorate the congestion via packaging. Packaging is unlikely to hurt in any environment. The induced latency associated with packaging is usually small compared with the latencies experienced on a congested network. Moreover, our experiments show that often the end-to-end latency is reduced when packaging is used. Packaging multiple media frames into a network message usually gives very

efficient packing of media frames into packets, even on networks with small MTUs. With very small MTUs, the effect of packaging may be limited, but at worst (if using two-dimensional adaptation), the results will be comparable to a pure bit rate scaling algorithm. Of course, packaging must be used within the guidelines of the transmission control framework or the induced latency may affect the quality of the delivered conference. Also, packaging must be done with the knowledge of the maximum acceptable buffer sizes at the sender. Packaging is very useful when adapting to access constraints, but should only be used when needed. It is perhaps tempting to always package at the minimum possible message rate, but this is not a good idea. When congestion is low, reducing the packaging (increasing the message rate by sending fewer media frames per message) reduces the induced latency and is likely to reduce the overall latency of the media stream. When congestion is low, induced latency can be greater than network latency. The importance of the packaging strategy increases with the number of multi-access networks in the conference transmission path and the load on these networks. Multi-access networks increase the potential for experiencing access constraints.

It is usually possible to package multiple audio frames in a single network packet, so audio packaging is usually effective. When possible, one should use small audio samples. Sending the samples in separate packets will result in low audio latency. When congestion increases, the conference may be forced to package the audio into larger messages. Video frames are typically much larger than audio frames and it is not usually possible to pack multiple video frames into a single message. As a rule of thumb, we suggest that if video frames are on average greater than 2.5K bytes, do not try to pack multiple video frames per message. In this case, the packets on most networks cannot carry more than 1 full video frame. Of the common network technologies, only 16 Mbits/second token rings have MTUs large enough to get more than one video frame in a packet. It is likely the conference will have to use some form of scaling to reduce the number of packets carrying video data.

In responding to an access constraint, a conferencing system that employs only bit rate scaling must scale the media streams enough to force a significant change in the number of generated packets. If the scaling does not significantly reduce the number of generated packets, the scaling is useless. If the bit rate reductions do not force reductions in the realized packet rate on the congested hop, the scaling will have little effect on the congestion. Indeed, scaling too little is worse than not scaling at all since

the scaling has little effect on the congestion and reduces the quality of the media frames that do reach the destination.

To limit the effects of access constraints, postpone fragmentation as late as possible. For example, in TCP/IP networks, it is common for the logical MTUs in routers to be set to default values such as 1,500 bytes even when the attached network supports much larger MTUs. Some TCP/IP implementations routinely use an MTU of 576 bytes if the destination is not on the local network. The intent of these schemes is to avoid the overhead associated with fragmentation [65], but both of these situations unnecessarily expose media streams to access constraints that could be alleviated by using the largest available packet sizes. Similarly, protocols, such as APPN, that use the smallest MTU in the path on all hops unnecessarily expose the stream to access constraints.

If the network is heavily congested, we recommend favoring audio performance over video performance. Audio quality is more important than video quality in the overall conference quality so it only makes sense to favor audio. Our rule of thumb is that if the delivered video frame rates are greater than 15 frames per second, we favor lower audio latency over increased video frame rates. This is done by raising the audio message rate before raising the video bit or message rates. On the other hand, if the delivered video rates fall below 15 frames per second, we do not increase the audio message rate until video performance has improved. This is the *poor video* heuristic.

We favor fidelity over latency up to a latency threshold. We generally use a 250 millisecond latency threshold. When latency is below 250 milliseconds, we favor increasing the bit rate over reducing the stream latency. When latency is over the threshold, we focus on reducing stream latency over increasing the stream fidelity. We base this rule of thumb on the assumption that differences in stream latencies are hard for people to detect up to some threshold. However, above that threshold, latency becomes a dominant factor in the perceived conference quality. The actual threshold employed may differ with different applications or systems. This heuristic is encompassed in the computation of EST_s where any latency below L_s^{max} is acceptable.

When access constraints are so severe that all operating points experience audio loss, consider introducing redundancy into the audio stream. One technique to do this is to retransmit each audio frame in several consecutive messages. The potential benefit of redundant audio frames on the audio gap rate may be significant and the overhead for

redundant audio is low. In particular, audio frames are relatively small and sending redundant audio will probably not affect the packet rate. Along the same lines, it is important to couple any transmission control scheme with a good display policy to handle the inevitable network jitter [103].

When designing a network to support both non-conference and conferencing traffic, with all other things being equal, favor a network with a large MTU over one with a small MTU. Networks with large MTUs have more potential for packaging than networks with small MTUs. Similarly, all other things being equal, choose a network with higher transmission speeds over one with slower speeds. If larger MTUs and higher speed are in conflict, the choice is more difficult. In general, we must decide whether to favor a faster network or a larger MTU on a case-by-case basis using the capacity and access constraint equations. Higher network speeds may reduce the maximum holding time of any particular node on the network and thus may limit the effect of access constraints by lowering the average medium access time. On the other hand, depending on the protocol used to acquire control medium access, higher network speeds may not lower the average medium access time. Larger MTUs may reduce the number of times a media stream must access the network, but perhaps at the expense of longer medium access waits due to the longer holding times of competing nodes. It is often difficult to precisely determine the effect of changing a network's speed or MTU because we do not know the relationships between the network speed, the MTU size, and the offered network traffic. With higher speed networks, data sources may or may not send more data. Applications using the network may exploit a larger MTU or they may not. Determining the effect of specific network changes may require significant knowledge of the network, the network traffic, and the applications and people using the network.

7.6. Best-effort and Reservation-based Systems

We have demonstrated that best-effort end-to-end techniques can deliver good quality conferences even in hostile environments. Obviously, there are traffic loads where best-effort techniques cannot succeed, but it is debatable whether reservation techniques can do any better. It is often difficult to orchestrate resource reservation throughout a network and it is also often difficult to implement the necessary service disciplines at each network element that are required in order to provide guaranteed performance. For resource reservation to be considered, it must provide significant

benefits over best-effort techniques, which tend to be simpler and require fewer changes to the existing infrastructure. However, resource reservation is not an option on some networks. For example, the nature of the CSMA/CD access protocol used by Ethernet makes it impossible to guarantee network performance. Clearly in such environments best-effort techniques provide a better solution than no solution at all. Even on networks where resource reservation is an option, resource reservation does not necessarily outperform best-effort techniques. For example, consider the access constraint experiments on token ring networks (Chapter 5). In these experiments, RS delivered very high quality conferences even when the token ring was over 99% utilized and the average medium access times where measured in tens of milliseconds. The network was completely utilized and the conference quality was near perfect. Reserving resources in this environment is very unlikely to produce significantly better results. It is also possible that resource reservation service disciplines will have a negative impact on the performance of the non-real-time streams.

It can be argued that the value of resource reservation is demonstrated only when the network is over-committed. The reservations ensure that network resources are allocated to the most important traffic and that less important traffic is barred from the network. The implicit assumption of this argument is that video conference traffic is the most important traffic on the network. This is a debatable assumption. Video conference traffic may be the *least* important traffic on the network. We predict that if video conferencing leads to consistent and significant over-commitment of resources in production networks, video conferencing will be the first application banned from the network. The ability to send data across computer networks is a fundamental requirement for most organizations and video conferencing data is likely the least valuable data transmitted. We also claim that most uses of video conferencing, with some notable exceptions, do not require perfect quality and that people are unwilling to pay the price for higher quality conferences. It is undoubtedly true that to *guarantee* very high quality conferences requires resource reservation. The question is whether there are many situations where guaranteed high quality is a requirement versus a desirable attribute. For most conferencing applications, we claim guaranteed performance will usually only be required for at most the audio stream and even then only at some relatively low resource requirement. While reservation-based techniques clearly have a place, we claim there will also be a continued need for best-effort techniques in the future. Best-effort service will always be the default and there will always be a cost incentive to use it.

7.7. Future work

Although we have gotten good results using the transmission control framework and the Recent Success algorithm in a number of environments, there are several opportunities to extend or improve these results. This section discusses a few of the items being considered for future work.

7.7.1. Integrate Recent Success into a Commercial Conferencing System

The conferencing system used in the experiments in this dissertation is an experimental system used only for research. Open issues symmetry of adaptations in two-way conferences and the performance of our framework with other sets of operating points. We would like to integrate the algorithm into a full-functioned conferencing system. The members of the Distributed Real-Time (DiRT) research group are working with Intel Corporation to integrate the Recent Success algorithm into the Intel ProShare video conferencing product. This effort is not yet complete, but we have built a prototype and the early results are promising [24, 79].

7.7.2. Improve Trigger and Selection Criteria

All tests using the Recent Success algorithm in this document used the same threshold parameters. These thresholds were originally set for a network composed only of token ring segments. It is likely that a different set of parameters would lead to better performance on other networks such as a network composed of Ethernet segments. Throughout our experiments we have kept the same thresholds for consistency, but it is possible that one or more of the parameters could be statically or dynamically changed based on network topology or traffic load. For example, one of these parameters is the 250 *ms* guideline for the maximum audio latency, L_a^{max}. This value is used in the calculation of the set of feasible operating points, $FOP_S(t)$, or more precisely the estimated feasible operating points, $EST_S(t)$. Note that for many of the experiments in Chapters 5 and 6, the average network latency for audio, the average delivered latency minus the induced latency, is lower when using the Recent Success algorithm than with any other algorithm, but that the average delivered latency is sometimes higher. The difference is due to the induced latency sometimes experienced when using the Recent Success algorithm. The value of L_a^{max} has a direct effect on the range of acceptable induced latencies, so if lower induced latencies are desired, the L_a^{max} variable could be set to a lower value. This would eliminate some of the

operating points where large numbers of audio frames are packed into a single message and lower the induced latency.

As another example, note that in some of the experiments in Chapters 5 and 6, Recent Success sometimes gives lower delivered video frame rates than other algorithms when the network is heavily congested. Part of the reason for this behavior is that in these experiments RS was given a set of video operating points that included operating points with every message rate (and corresponding frame rate) rate from 1 to 30 messages per second (although in Chapter 6 we eliminated the points below 5 messages per second). Since the operating points were defined, the Recent Success algorithm could choose video operating points with very low frame rates when the network was heavily congested. If video frame rates under 15 frame/second are considered unacceptable (for example), the operating points associated with these frame rates may be removed. This removes them as candidate operating points and RS would be forced to use operating points producing frame rates over 15 frames per second.

We have not investigated the sensitivity of particular parameters such as the success and failure thresholds in the Recent Success state machine. We also have not investigated dynamically changing parameters or customizing parameters for particular network technologies or topologies. It seems likely that the performance of Recent Success could be tuned for a particular environment or application by adjusting the algorithm parameters.

7.7.3. Improve Feedback Scheme

When packaging large numbers of audio frames per message, there may be situations where n messages are received one feedback interval and $n + 1$ messages are received in the next interval. Since many audio frames are packaged per message, the delivered audio frame rate in the two consecutive intervals may differ greatly and prevent the algorithm from recognizing an equilibrium state. This prevents the algorithm from probing the network for higher performance operating points. We could potentially improve the algorithm by averaging the frame deliveries over consecutive feedback intervals at the conference destination.

7.7.4. Use Coordinated Stream Control

All the experiments in this dissertation control the audio and video streams separately (except for the use of the *poor video* heuristic). It is possible that we could improve the delivered quality of the conference by combining the audio and video operating points into a single stream and representing the capabilities of this stream with a single set of operating points. By considering audio and video as one stream, the system designer may be able to identify combinations of audio and video that provide better perceived quality than if the operating points for the streams are picked independently. The transmission control policy would pick only a single operating point that defined generation and transmission parameters for both audio and video.

When we first started this research, we expected consolidated management of the streams to outperform independent control. Unexpectedly, independent control worked so well that we have never investigated consolidated control. The potential benefits of consolidated control seem small given the level of success achieved with independent control, but it remains an option to be investigated.

7.7.5. Consider Interaction of Multiple Simultaneous Conferences

All the experiments in this dissertation had only one video conference application and many non-conferencing applications sharing the network. We have not investigated the effect of multiple simultaneous conferences, all adapting with independent knowledge and timing, on the delivered quality of the conferences.

7.7.6. Evaluate the Effects of Conferences on Non-conference Traffic

We have not evaluated the effect video conferencing streams have on the performance of non-conference traffic such as FTP transfers and HTTP traffic. It is important to investigate the relative performance of non-conference traffic when different transmission control algorithms are used to transmit a conference. Given the low levels of loss with the Recent Success conferences, we hypothesize that conferences using Recent Success will have less adverse affect on non-conference traffic than non-adaptive conferences or conferences that scale on the video bit rate. Our experiments imply that conferences using Recent Success reduce their transmission rates to only what the network can sustain. This is not the case with the other algorithms, as indicated by the high message loss seen in the BL, VSO, and TSO

experiments. Since Recent Success does a better job at not overloading the network with excess traffic, we expect conferences using Recent Success with have less impact on other traffic than the other transmission control schemes.

7.7.7. Add Multicast Support

All the conferences in this dissertation were point-to-point conferences. We have not yet attempted to extend the transmission control framework to support multicast conferences. Others have worked with feedback systems in multicast environments. For example, IVS, a video conferencing application developed at INRIA, has a scheme for collecting feedback from multiple receivers in one-way conferences [14]. The hard part of building best-effort systems in a multicast environment is figuring out how to adjust the individual streams. IVS uses a relatively simple decision based on the fraction of receivers receiving poor video, but they are investigating other techniques such as hierarchical scaling. We have not investigated multicast support with Recent Success.

7.7.8. Extend Adaptations into Routers

The focus of this paper has been on end-to-end techniques, but there is potentially a great opportunity for hop-by-hop adaptations, particularly when addressing access constraints on multi-access networks. For example, Roussos, *et al*, applied some packaging techniques to intermediate routers for audio support [95]. We have begun using the transmission control framework and associated media packaging and scaling adaptations in intermediate routers. The preliminary results are promising [62].

7.7.9. The Framework and Long Network Paths

Theoretically, the description of the transmission control framework and the effect of capacity and access constraints are applicable to network paths of any length. However, all our experiments have been on networks where the path from source to destination was less than six hops. We are considering experiments to evaluate the scalability of the techniques presented here to long network paths. Based on our preliminary results, we expect to find that the techniques described here do scale to large networks [79].

7.6.10. Consider Perceptual Effects of Scaling and Latency Changes

Some have questioned the perceptual effects of dynamically changing the fidelity and latency of the conference. They question whether people using the system will find the changes disconcerting, annoying, or confusing. This is a legitimate question and worthy of investigation. However, the techniques used by the transmission control framework and Recent Success to change the coding schemes, frame rates, or latency are no different from the techniques used by systems employing video scaling techniques or buffered display policies and we contend that the techniques described here are no more or less intrusive or objectionable than current accepted practices. This conjecture needs to be confirmed with experiments.

7.6.11. ATM

Much of the current discussion of audio and video support on computer networks has focused on ATM. ATM is a very interesting technology since it may simultaneously remove most capacity constraints through its support for very high transmission speeds and access constraints through switching. ATM may eliminate many latency issues due to fast transfer speeds and limited buffering in the switches. Furthermore, the problems associated with very high speed networks and feedback-based flow control [111] are less applicable to conferencing systems since data is consumed at the same rate it is produced. Thus, conferencing systems are amenable to rate-based control. Video conferencing applications are only likely to experience problems on an ATM network if the aggregate data rates of all data sources is greater than the switch's capacity.

Unfortunately, ATM is a relatively new technology and we have little information about video conferences actually running on ATM networks. Constant bit rate (CBR) schemes have been proposed for ATM and continuous media can often be smoothed to match CBR rates; however, audio and particularly video are fundamentally variable bit rate streams. Unfortunately, as of 1995, the ATM variable bit rate (VBR) service was not defined [100]. Real-time techniques for guaranteed delivery of constant bit rate streams are in some sense direct extensions of real-time scheduling theory and seem feasible, but real-time support for variable bit rate services is more complicated. For example, variable bit rates make specification of traffic descriptors for media streams difficult [111]. Furthermore, implementing support for fractional guarantees rather than deterministic guarantees is an ongoing research problem [33]. ATM cell

sizes become an issue when cell loss is possible because ATM cell sizes are not optimal for video [111]. The effect of single cell loss can be greatly magnified if it makes a full media frame unplayable [111].

It is likely that the first widespread use of ATM technology will be as a switched backbone to existing local area networks [111]. Most computers will not be directly connected to an ATM switch. In this mixed environment, video conferences are likely to experience the same constraints present on existing networks without ATM. The ATM switch or set of interconnected switches function essentially as a very fast router between existing networks. When the data leaves the source node, it must cross some existing LAN technology and when the data leaves the switch, the data must again cross some existing LAN. The implication is that video conferences on such networks are likely to experience many of the same access constraints as on existing networks. If the ATM switches replace an existing conventional LAN backbone, access constraints experienced on the old backbone are likely to be eliminated, but access constraints on the connecting LANs will still exist. It seems likely that for performance reasons messages transmitted over ATM backbones will be reassembled before they are placed on the outbound LANs [101]. If so, then end-to-end message rate manipulation is still viable and valuable. Capacity constraints within existing routers are likely to be eliminated by the high-speed ATM switches, but capacity constraints are typically due to relatively low-speed wide-area connections rather than in existing routers. We are in the process of installing a set of ATM switches at the University of North Carolina at Chapel Hill to begin our investigation of the actual effects of ATM on video conferencing.

References

[1] 3Com/Microsoft, *3Com/Microsoft LAN Network Driver Interface Specification*, Version 2.0.1 Final, October 8, 1990.

[2] Ahuja, V., Routing and Flow Control in Systems Network Architecture, IBM Systems Journal, 18, pp. 298-314.

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

[4] Anderson, D., *Metascheduling for Continuous Media*, ACM Transactions on Computer Systems, Vol. 11, No. 3, August, 1993, pp. 226-252.

[5] Aravind, R., Cash, G., Duttwiler, D., Hang, H., Haskell, B., Puri, A., *Image and Video Coding Standards*, AT&T Technical Journal, Jan-Feb, 1993, p. 67-89.

[6] Asimov, I., *Understanding Physics: Motion, Sound, and Heat*, New American Library, Inc., New York, April, 1966.

[7] Banerjea, A., Knightly, E., Templin, F., Zhang, H., *Experiements with the Tenet Real-Time Protocol Suite on the Sequoia 2000 Wide Area Network*, Proceedings of ACM Multimedia '94, San Francisco, CA, October, 1994, p. 183-191.

[8] Banerjea, A., Ferrari, D., Mah, B., Moran, M., Verma, D., Zhang, H., *The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences*, University of California at Berkeley, TR-94-059, November, 1994.

[9] Basile, C., Cavallerano, A.P., Deiss, M.S., Keeler, R., Lim, J.S., Luplow, W.C., Paik, W.H., Petajan, E., Rast, R., Reitmeir, G., Smith, T.R., Todd, C., *The U.S. HDTV Standard: The Grand Alliance*, IEEE Spectrum, April, 1995, pp. 36-45.

[10] Bauerfield, W., Westbrock, H. *Multimedia Communication with High-Speed Protocols*, Computer Networks and ISDN Systems, vol. 23, 1991, p. 143-151.

[11] Bellamy, J., *Digital Telephony*, John Wiley & Sons, ISBN 0-471-08089-6, 1982, p. 83-479.

[12] Bertsekas, D., Gallager, R., *Data Networks, Second Edition*, Prentice Hall, Englewood Cliffs, NJ, ISBN 0-13-200916-1, 1992.

[13] Boggs, D., Mogul, J., Kent, C., *Measured Capacity of an Ethernet: Myths and Reality*, Computer Communications Review, vol. 18, no. 4, Proceedings of the ACM SIGCOMM '88 Workshop, August 1988, pp. 222-234.

[14] Bolot, J., Turletti, T., *A Rate Control Mechanism for Packet Video in the Internet*, Proceedings of IEEE INFOCOMM '94, Toronto, Canada, June 1994, pp. 1216-1223.

[15] Bolot, J-C., Turletti, T., Wakeman, I., *Scalable Feedback Control for Multicast Video Distribution in the Internet*, Proceedings of ACM SIGCOMM, 1994, pp. 58-67.

[16] Buford, J.F.K. (ed.), *Multimedia Systems*, Addison-Wesley, Reading, MA, ISBN 0-201-53258-1, 1994.

[17] Bunzel, M. J., Morris, S. K., *Multimedia Applications Development Using DVI Technology*, McGraw-Hill, ISBN 0-07-043297-X, 1992.

[18] Chakrabarti, S., Wang, R., *Adaptive Control for Packet Video*, Proceedings of IEEE International Conference on Multimedia Computing and Systems 1994, Boston, MA, May 1994, pp. 56-62.

[19] Ciciora, W.S., *Inside the Set-top Box*, IEEE Spectrum, April, 1995, pp. 70-75.

[20] Comer, D., *Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architecture*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[21] Cormen, T.H., Leiserson, C.E., Rivest, R.L., *Introduction to Algorithms*, MIT Press, Cambridge, MA, ISBN 0-26203141-8, 1990, p. 337-345.

[22] Creative Labs, *Sound Blaster 16 Users Guide*, Creative Labs, Inc., 1995.

[23] Crouch, P., Hicks, J., Jetzt, J., *ISDN Personal Video*, AT&T Technical Journal, Jan-Feb, 1993, p. 33-40.

[24] Danneels, G., Intel Architecture Labs, Hillsboro, OR, *gunner_danneels@ccm.jf.intel.com*, personal communication, January, 1996.

[25] Delgrossi, L., Halstrick, C., Herrtwich, R., Stuttgen, H., *HeiTP - A Transport Protocol for ST-II*, IEEE Globecom, '92, Orlando, 1992.

[26] Delgrossi, L., Halstrick, C., Hehmann, D., Herrtwich, R., Krone, O., Sandvoss, J., Vogt, C., *Media Scaling for Audiovisual Communication with the Heidelberg Transport System*, Proc. ACM Multimedia 93, Anaheim, CA, August 1993, pp. 99-104.

[27] Delgrossi, L., Herrtwich, R., Vogt, C., Wolf, L., *Reservation Protocols for Internetworks: A Comparison of ST-II and RSVP*, Proceedings of the Fourth International Workshop on Network and Operating System Support for Digital Audio and Video (eds. D. Shepherd, G. Blair, G. Coulson, N. Davies, F. Garcia), Lancaster, UK, November 1993, Springer-Verlag, pp. 195-203.

[28] Denning, P.J., Buzen J.P., *The Operational Analysis of Queueing Network Models*, ACM Computing Surveys, Vol. 10, No. 3, September 1978, pp. 225-261.

[29] Dupuy, S., Tawbi, W., Horlait, E., *Protocols for High-Speed Multimedia Communications Networks*, Computer Communications, vol. 15, no. 6, July/August 1992, pp.349-358.

[30] Edwards, F.R., Schultz, M.F., *A Priority Media Access Control Protocol for Video Communication Support on CSMA/CD LANs*, ACM Multimedia Systems, vol. 2, no. 6, January, 1995, pp. 243-255.

[31] Eriksson, H., *MBONE: The Multicast Backbone*, Communications of the ACM, vol. 37, no. 8, August, 1994, pp. 54-60.

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

[33] Ferrari, D., Banerjea, A., Zhang, H., *Network Support for Multimedia: A Discussion of the Tenet Approach*, Computer Networks and ISDN Systems, Vol. 26, No. 10, July 1994, pp. 1267-1280.

[34] Fish, R., Kraut, R., Root, R., Rice, R., *Video as a Technology for Informal Communication*, Communications of the ACM, vol. 36, No. 1, January 1993, pp. 48-61.

[35] Fowler, H., Leland, W.E., *Local Area Network Traffic Characteristics with Implications for Broadband Network Congestion Management*, IEEE Journal on Selected Areas of Communications, vol.9, 1991, pp. 1139-1149.

[36] Fox, B., *The Digital Dawn in Europe*, IEEE Spectrum, April, 1995, pp. 50-53.

[37] Frederick, B, *nv*, Manual Pages, Xerox Paolo Alto Research Center, Palo Alto, CA.

[38] George, F., Young, G., *SNA Flow Control: Architecture and Implementation*, IBM Systems Journal, 21, pp.179-210.

[39] Gerla, M., Kleinrock, L., *Flow Control: A Comparative Survey*, IEEE Transactions on Communications, vol. 28, no. 4, April, 1980, pp. 553-574.

[40] Golestani, S., *A Stop-and-Go Framework for Congestion Management*, Proceedings of SIGCOMM '90, August, 1990, pp. 8-18.

[41] Green, J., *The Evolution of the DVI System Software*, Communications of the ACM, vol. 35, no. 1, January, 1992, p. 53-67.

[42] Gruber, J., Le, N., *Performance Requirements for Integrated Voice/Data Networks*, IEEE Journal on Selected Areas in Communications, vol. SAC-1, no. 6, December, 1983, pp. 981-1005.

[43] Haas, Z., *Adaptive Admission Congestion Control*, ACM SIGCOMM-Computer Communications Review, vol.5, October 1991, pp. 58-76.

[44] Harney, K., Keith, M., Lavell, G., Ryan, L.D., Stark, D.J., *The i750®Video Processor: A Total Multimedia Solution*, Communications of the ACM, vol. 34, no. 4, April, 1991, p. 64-78.

[45] Herrtwich, R., *Time Capsules: An Abstraction for Access to Continuous-Media Data*, Proceedings of 11th Real-Time Systems Symposium, December 1990, pp. 11-20.

[46] Hoffman, D., Speer, M., Fernando, G., *Network Support for Dynamically Scaled Multimedia Data Streams*, Proceedings of the Fourth International Workshop on Network and Operating System Support for Digital Audio and Video (eds. D. Shepherd, G. Blair, G. Coulson, N. Davies, F. Garcia), Lancaster, UK, November 1993, Springer-Verlag, pp.240-251.

[47] Huynh, L., Chang, R., *ARB - An Adaptive Rate Based Flow/Congestion Managment Scheme for High-Speed Networks*, IBM Corporation, unpublished document, 1993.

[48] IBM, *Local Area Network Technical Reference,* IBM Corporation, Fourth Edition, 1990.

[49] INRIA, ftp://zenon.inria.fr (138.96.32.21) in /rodeo/ivs.

[50] Intel, *Intel ProShare Personal Conferencing Video System 200*, Intel Corporation, 1993.

[51] Isaacs, E., Tang, J.C., *What Video Can and Can't Do for Collaboration: A Case Study*, Proceedings of ACM Multimedia 93, Anaheim, CA, August, 1993, pp. 199-205.

[52] ISO/IEC, *Digital Compression and Coding of Continuous-Tone Still Images, Part 1: Requirements and Guidelines*, ISO/IEC JTC1 Committee Draft 10918-1 Feb., 1991.

[53] Jacobson, V., McCanne, S., *vat*, Manual Pages, Lawrence Berkeley Laboratory, University of California, Berkeley, CA., ftp://ftp.ee.lbl.gov.

[54] Jacobson, V., *Congestion Avoidance and Control*, Proceedings of ACM SIGCOMM '88, Stanford, CA, August, 1988, pp. 314-329.

[55] Jacobson, V., *Multimedia Conferencing on the Internet*, SIGCOMM '94 Tutorial, London, UK, August, 1994.

[56] Jain, R., *Congestion Control in Computer Networks: Issues and Trends*, IEEE Network, 4(3), May, 1990, pp. 24-30.

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

[58] Jeffay, K., Lin, J.K., Menges, J., Smith, F.D. and Smith, J.B., *Architecture of the Artifact-Based Collaboration Systems Matrix*, Proceedings of ACM CSCW '92, Toronto, Canada, 1992.

[59] Jeffay, K., *The Real-Time Producer/Consumer Paradigm: A Paradigm for the Construction of Efficient, Predictable Real-Time Systems,* Proceedings of ACM/SIGAPP Symposium on Applied Computing, Indianapolis, IN, February, 1993, pp. 796-804.

[60] Jeffay, K., Stone, D., Talley, T., Smith, F.D., *Adaptive, Best-Effort Delivery of Digital Audio and Video Across Packet-Switched Networks,* Proceedings of the Third International Conference on Network and Operating System Support for Digital Audio and Video, La Jolla, CA, V. Rangan (Ed.), Lecture Notes in Computer Science, vol. 712, November, 1993, pp. 3-14.

[61] Jeffay, K., Stone, D., and Smith, F.D., *Transport and Display Mechanisms for Multimedia Conferencing Across Packet-Switched Networks,* Computer Networks and ISDN Systems, Vol. 26, No. 10, July 1994, pp. 1281-1304.

[62] Jeffay, K., Parris, M., Smith, F.D., Talley, T., *A Router-Based Congestion Control Scheme For Real-Time Continuous Media*, Sixth International Workshop on Network and Operating System Support for Digital Audio and Video, Lecture Notes in Computer Science, Springer-Verlag, Zushi, Japan, April 1996.

[63] Johnson, J.T., *Videoconferencing: Not Just Talking Heads*, Data Communications, November, 1991, pp. 66-88.

[64] Kanakia, H., Mishra, P., Reibman, A., *An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport,* Proceedings on ACM SIGCOMM '93, September, 1993, pp. 20-31.

[65] Kent, C., Mogul, J., *Fragmentation Considered Harmful*, Proceedings of Frontiers in Computer Communications Technology, ACM SIGCOMM '87, Stowe, Vermond, August, 1987, pp. 390-401.

[66] Kolman, B., *Introductory Linear Algebra with Applications*, Macmillan Publishing Co., Inc., New York, NY, ISBN 0-02-365950-5, 1976.

[67] Kurose, J., *Open Issues and Challenges in Providing QoS Guarantees in High Speed Networks*, Computer Communications Review, vol.23, no.1 , January, 1993, pp. 6-15.

[68] Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K.C., *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*, Prentice-Hall, Inc., Englewood Cliffs, NJ, ISBN 0-13-746975-6, 1984.

[69] Le Gall, D., *MPEG: A Video Compression Standard for Multimedia Applications*, Communications of the ACM, vol. 34, no. 4, April, 1991, p. 46-58.

[70] Leland, W.E., Wilson, D.V., *High Time-resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection*, Proceedings of IEEE INFOCOM, 1991, p. 1360-1366.

[71] Leland, W.E., Willinger, W., Taqqu, M.S., Wilson, D.V., *On the Self-Simlar Nature of Ethernet Traffic*, IEEE Network, , September, 1993, p. 183-193.

[72] Levy, S., *Calling All Computers*, Newsweek, May 13, 1996, p. 43-46.

[73] Liou, M., *Overview of the px64 kbits/s Video Coding Stnadard*, Communications of the ACM, vol. 34, no. 4, April, 1991, p. 59-63.

[74] Lippman, A., *Feature Sets for Interactive Images*, Communications of the ACM, vol. 34, no. 4, April, 1991, p. 92-102.

[75] Mercer, C., Savage, S., Tokuda, H., *Processor Capacity Reserves: Operating System Support for Multimedia Applications*, IEEE International Conference on Multimedia Computing and Systems, Boston, MA, May 1994, pp. 90-99.

[76] Natarajan, K., *Video Servers Take Root*, IEEE Spectrum, April, 1995, pp. 66-69.

[77] Naylor, W.E., Kleinrock, L., *Stream Traffic Communication in Packet-switched Networks: Destination Buffering Considerations*, IEEE Transactions on Communications, vol. 12, 1982, pp. 2527-2534.

[78] Nee, P., University of North Carolina at Chapel Hill, Chapel Hill, NC, *nee@cs.unc.edu*, personal communication, February, 1995.

[79] Nee, P., Jeffay, K., Danneels, G., *The Performance of Two-Dimensional Media Scaling for Internet Videoconferencing: Experiments with ProShare on the Internet*, Seventh International Workshop on Network and Operating System Support for Digital Audio and Video, to appear, 1995.

[80] Noll, A.M., *Anatomy of a Failure: Picturephone Revisited*, Telecommunications Policy, May/June, 1992.

[81] Nomura, M., Fuji, T., Ohta, N., *Basic Characteristics of Variable Rate Video Coding in ATM Environment*, IEEE Journal on Selected Areas of Communication, vol. 7, 1989, p. 752-760.

[82] Ozer, J., *Indeo 3.2: The New (Iteration) Codex on the Block*, CD-ROM Professional, vol. 7, no. 6, November, 1994, p. 70.

[83] Pan, D. Y., *Digital Audio Compression,* Digital Technical Journal, Vol. 5, No. 2, Spring 1993, pp. 28-40.

[84] Partridge, C., Pink, S., *An Implementation of the Revised Internet Stream Protocol (ST-II),* Second International Workshop on Network and Operating Systems Support for Digital Audio and Video, Heidelberg, Germany, November 1991.

[85] Partridge, C., *Editors Note: The End of Simple Traffic Models,* IEEE Network, , September, 1993, p. 3.

[86] Pennebaker, W., Mitchell, J., *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, ISBN 0-442-01272-1, 1993.

[87] Petzold, C., *The Multimedia Extensions for Windows - Enhanced Sound and Video for the PC,* Microsoft Systems Journal, vol. 6, no. 2, March, 1991, pp. 19-26.

[88] Pizer, S.M., Wallace, V.L., *To Compute Numerically: Concepts and Strategies*, Little, Brown and Company, Boston, MA, ISBN 0-316-70940-9, 1983.

[89] Poole, L., *Quicktime in Motion*, Macworld, September, 1991, pp. 154-159.

[90] Ramakrishnan, K., Jain, R., *A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer*, Proceedings of ACM SIGCOMM, Stanford, CA, August 1988, pp. 303-313.

[91] Rangan, P. V., Vin, H. M., *Designing File Systems for Digital Video and Audio*, Proceedings of ACM Operating Systems Review, vol. 25, no. 5, October 1991, pp. 81-94.

[92] Rao, K.R., Yip, P., *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, Harcourt Brace Janovich, ISBN 0-12-580203-X, 1990.

[93] Ripley, G. D., *DVI - A Digital Multimedia Technology*, Communications of the ACM, v. 32, no. 7, July, 1989, p. 811-822.

[94] Rodriguez, A., Morse, K., *Evaluating Video Codecs*, IEEE Multimedia, vol. 1, no. 3, Fall, 1994, p. 25-33.

[95] Roussos, J., Economou, E., Philokyprou, G., *Congestion Control Protocols for Interconnected LANs Supporting Voice and Data Traffic*, Computer Communications Review, vol. 17, no. 1, January, 1994, p. 25-34.

[96] Schulzrinne, H., *Voice Communication Across the Internet: A Network Voice Terminal*, Technical Report, University of Massachusetts, 1992.

[97] Schulzrinne, H., Casner, S., *RTP: A Real-Time Transport Protocol*, Internet Draft, July 30, 1993.

[98] Sellen, A., *Speech Patterns in Video-Mediated Conversations*, Proceedings of CHI '92, Monterey, CA, May 1992, pp. 49-59.

[99] Sijstermans, F., van der Meer, J., *CD-I Full-Motion Video Encoding on a Parallel Computer*, Communications of the ACM, vol. 34, no. 4, April, 1991, p. 81-91.

[100] Siu, K-Y., Jain, R., *A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management*, Computer Communication Review, vol. 25, no. 2, April, 1995, p. 6-20.

[101] Smith, F.D., University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, *smithfd@cs.unc.edu*, personal communication, April, 1997.

[102] Steinmetz, R., Nahrstedt, K., *Multimedia: Computing, Communications, and Applications*, Prentice Hall, Upper Saddle River, NJ, ISBN 0-13-324435-0, 1995.

[103] Stone, D.L., *Managing the Effect of Delay Jitter on the Display of Live Continuous Media*, Ph.D. dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, May, 1995.

[104] Stone, D.L., Jeffay, K., *An Empirical Study of Delay Jitter Management Policies*, ACM Multimedia Systems, vol. 2, no. 6, January, 1995, pp. 267-279.

[105] Talley, T.M., Jeffay, K., *Two-Dimensional Scaling Techniques for Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams*, Proceedings of ACM Multimedia '94, San Francisco, CA, October, 1994, pp. 247-254.

[106] Tannenbaum, A.S., *Computer Networks*, Prentice-Hall, Inc., Englewood Cliffs, NJ, ISBN 0-13-162959-X, 1989.

[107] Tawbi, W., Horn, F., Horlait, E., Stefani, B., *Video Compression Standards and Quality of Service*, The Computer Journal, vol. 36, no. 1, 1989, pp. 43-54.

[108] Topolcic, C., *Experimental Internet Stream Protocol, Version 2 ST-II*, RFC 1190, October, 1990.

[109] Trajkovic, L., Golestani, S., *Congestion Control for Multimedia Services*, IEEE Network Magazine, vol. 6, no. 5, 1992, pp. 20-25.

[110] Turletti, T., *H.261 Software Codec for Videoconferencing Over the Internet*, INRIA Technical Report 1834, January, 1993.

[111] Vetter, R., *ATM Concepts, Architectures, and Protocols*, Communications of the ACM, vol. 38, no. 2, February, 1995, p. 30-38.

[112] Wakeman, I., *Packetized Video - Options for Interaction Between the User, the Network and the Codec*, Computer Journal, vol. 36, 1993, pp. 55-67.

[113] Wallace, G., *The JPEG Still Picture Compression Standard*, Communications of the ACM, vol. 34, no. 4, April, 1991, p. 30-44.

[114] Wang, K.S., Normile, J.O., Wu, H., Rodriguez, A.A., *Vector-quantization-based Video Code for Software-only Playback on Personal Computers*, Multimedia Systems, vol. 2, no. 5, December, 1994, p. 191-203.

[115] Weaver, K., University of North Carolina at Chapel Hill, Chapel Hill, NC, *weaver@cs.unc.edu*, personal communication, April, 1995.

[116] Wolf, C., *Video Conferencing: Delay and Transmission Considerations*, in Teleconferencing and Electronic Communications: Applications, Technologies, and Human Factors, L. Parker and C. Olgren (Eds.), 1982.

[117] Womble, S., IBM Integrated Systems Services Corporation (ISSC), Austin, TX, *womble@vnet.ibm.com*, personal communication, November, 1994.

[118] Zhang, L., Deering, S., Estrin, D., Shenker, S., Zappala, D., *RSVP: A New Resource ReSerVation Protocol*, IEEE Network, September, 1993.