AN ARCHITECTURE FOR DISTRIBUTED, INTERACTIVE, MULTI-STREAM, MULTI-PARTICIPANT AUDIO AND VIDEO

Brian K. Schmidt

Technical Report No.: CSL-TR-99-781

April 1999

This project is, in part, supported by funding from an NSF Young Investigator Award and by Sun Microsystems, Inc.

An Architecture for Distributed, Interactive, Multi-Stream, Multi-Participant Audio and Video

Brian K. Schmidt

Technical Report No.: CSL-TR-99-781

April 1999

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
William Gates Computer Science Building, 4A-408
Stanford, CA 94305-9040
<pubs@shasta.stanford.edu>

Abstract

Today's computer users are becoming increasingly sophisticated, demanding richer and fuller machine interfaces. This is evidenced by the fact that viewing and manipulating a single stream of full-size video along with its associated audio stream is becoming commonplace. However, multiple media streams will become a necessity to meet the increasing demands of future applications. An example which requires multiple media streams is an application that supports *multi-viewpoint audio and video*, which allows users to observe a remote scene from many different perspectives so that a sense of immersion is experienced. Although desktop audio and video open many exciting possibilities, their use in a computer environment only becomes interesting when computational resources are expended to manipulate them in an interactive manner. We feel that user interaction will also be an extremely important component of future multimedia systems, and the methods of interaction will become increasingly complex. In addition, future applications will make significant demands on the network in terms of bandwidth, quality of service guarantees, latency, and connection management.

Based on these trends we feel that an architecture designed to support future multimedia applications must provide support for several key features. The need for numerous media streams is clearly the next step forward in terms of creating a richer environment. Support for non-trivial, fine-grain interaction with the media data is another important requirement, and distributing the system across a network is imperative so that multiple participants can become involved. Finally, as a side effect of the network and multi-participant requirements, integral support for and use of multicast will be a prime architectural component. The goal of our work is to design and implement a complete system architecture capable of supporting applications with these requirements.

Key Words & Phrases: multimedia, network, 3D video, distributed, architecture, interactive, virtual reality, augmented reality.

1 Introduction

Traditional computer architecture research has focussed primarily on the design and performance of individual machines. With the increasing importance of network technology and multimedia interfaces, future architectures will span machine boundaries and will need to provide adequate support for time-critical operations. The work we propose seeks to investigate this new class of system architectures.

1.1 Motivation

Today's computer users are becoming increasingly sophisticated, demanding richer and fuller machine interfaces. This is evidenced by the fact that "multimedia" has become a trendy buzzword used to hype seemingly every new computer product to hit the marketplace. In response many researchers are investigating the best means to make desktop audio and video a reality, and this area is beginning to mature. The MPEG I and II coding standards now enjoy widespread acceptance, and numerous vendors market hardware implementations. There are still unresolved issues, e.g. synchronization, quality of service (QoS) guarantees, transport and storage mechanisms, etc. However, viewing and manipulating a single stream of full-size video along with its associated audio stream is becoming commonplace. This provides sufficient expressiveness for numerous applications, but many others will require a richer environment. Consequently, work has been done to investigate means for supporting high definition television (HDTV), as well as the use of two video streams to create three-dimensional television (3DTV). A natural extension is the need for numerous streams of audio and video. Multiple media streams will become a necessity to meet the increasing demands of future applications. An example which requires multiple audio and video streams is an application that supports *multi-viewpoint audio and video*. Stereo (3D) audio and video offer only a single perspective from which a scene may be viewed, but multi-viewpoint audio and video allow a user to observe a scene from many different perspectives so that a sense of immersion is experienced. Multi-viewpoint audio and video are composed of several different streams and thus represent a significant technical challenge over current desktop audio and video.

Although desktop audio and video open many exciting possibilities, the danger of relegating a high performance workstation to the role of an extremely expensive television set is quite real. Further, simple distribution and display of media data only allows for a one-way exchange of information, but interaction with the media opens up numerous possibilities for two-way exchanges. Thus, the use of audio and video in a computer environment only becomes interesting when computational resources are expended to manipulate them. This can be accomplished in many ways. For example, computer generated graphics can be mixed with the video, content-based indexing and retrieval can be utilized, or mechanisms for user interaction can be provided. We feel that user interaction will be an extremely important component of future multimedia systems, but the methods of interaction will become increasingly complex. Simple VCR-like controls represent trivial interactions. The next generation of multimedia applications will require much more fine-grained control over media streams, and a system to support the use of audio and video must provide the means for these types of manipulations to be performed.

Another interesting trend in modern computing is increasing network connectivity. Not only do networks allow resources to be distributed, but they also provide the opportunity for multiple users to simultaneously participate in a common enterprise. Future applications will make significant demands on the network in terms of bandwidth, quality of service guarantees, latency, and connection management. The emergence of fast packet-switching and fiber optic networks represent

enabling technologies for many such applications, and a great deal of research into this area has begun. In addition entire system architectures centered around networks are being built today, and more work is sure to continue in the future. The upshot of this trend is that forward-looking research projects must incorporate network technology into their design goals.

Based on these trends we feel that an architecture designed to support future multimedia applications must provide support for several key features. The need for numerous media streams is clearly the next step forward in terms of creating a richer environment. Support for non-trivial, fine-grain interaction with the media data is another important requirement, and distributing the system across a network is imperative so that multiple participants can become involved. Finally, as a side effect of the network and multi-participant requirements, integral support for and use of multicast will be a prime architectural component. The goal of our work is to design and implement a complete system architecture capable of supporting applications with these requirements.

1.2 System Overview

In response to the demands and trends outlined in the previous section, we propose to design and implement an architecture for distributing multiple streams of audio and video across a network to multiple participants who interact with the media at a fine-grained level. There are numerous applications which can benefit from such a system. For example, a security system in a large office building which employs computers to "view" numerous media streams will require the data to be transmitted to multiple hosts, all of which are switching between viewed streams at a high rate. Another example is a service that broadcasts many "channels" of audio and video to a large number of subscribers, similar to current cable television providers. Although each viewer will switch channels relatively slowly, the net effect over many viewers requires the network to support fine-grain routing changes.

We have chosen to utilize an application for viewing multi-viewpoint audio and video as the test case for the proposed system architecture. Figure 1 depicts an overview of this system. An array of cameras is used to capture numerous streams of video, providing multiple slightly displaced viewpoints from which a scene is observed. On the order of sixty-four cameras will be used, and each will also incorporate a microphone to acquire audio data. This generates a series of audio and video streams, which are then digitized, compressed, aggregated, and packetized for transmission across a network. The system design supports multiple participants simultaneously observing the scene from potentially different viewpoints. Interaction is supported in the form of perspective changes. Participants notify the system (e.g. through use of a head tracker) that a viewpoint change is desired, and the system responds in a timely fashion by presenting different streams to the user.

This multi-viewpoint audio and video viewing system is an excellent representative example of a class of applications that includes the next generation of features outlined above. Numerous media streams are being transmitted across a network to multiple participants. Different viewers receive different subsets of the streams and interact with them through viewpoint changes. These perspective changes are typically coupled with user head motion and must experience low latency to avoid user discomfort. Thus, switching between streams is a dynamic and fine-grain process which represents non-trivial interaction with the media data.

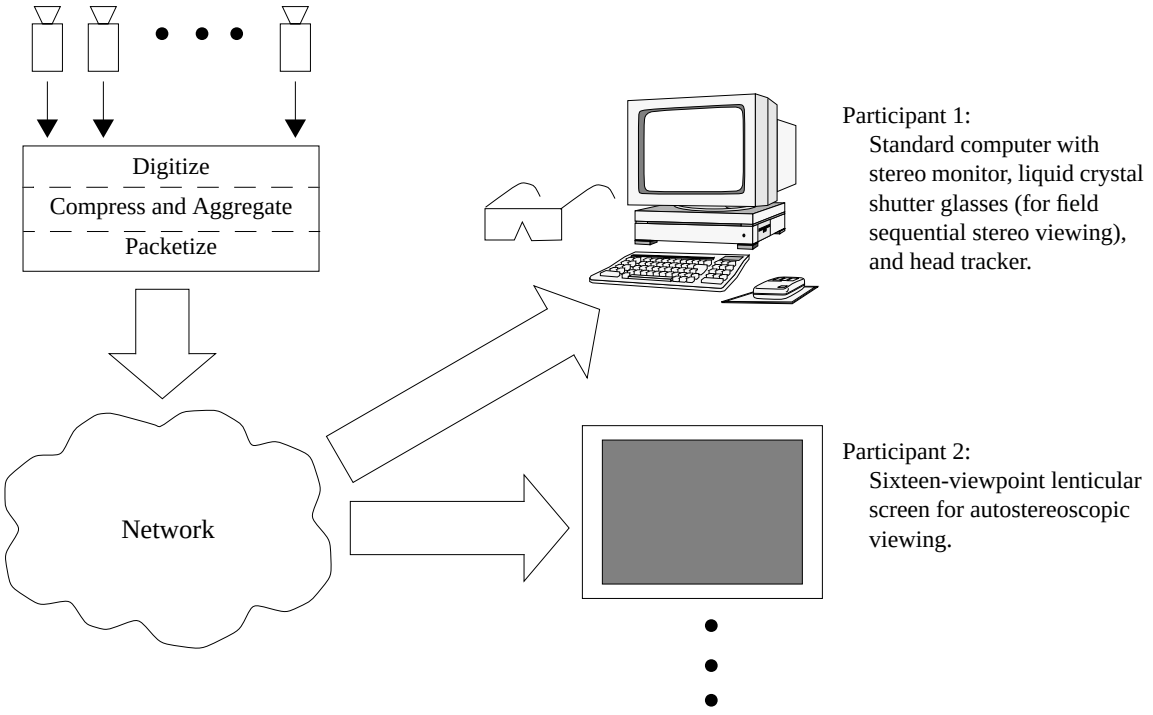

Figure 1 System overview.

1.3 Sample Applications

To obtain a sense of the potential impact of the multi-viewpoint audio and video application domain, it is useful to examine a few representative examples. One of the promising uses of this technology is in the area of medicine [79]. Such a system could be used to replace standard laparoscopic surgery, the use of small incisions through which long-handled instruments are inserted to perform the operation. Since the surgeon cannot see the tips of the instruments, a tiny video camera is inserted through an umbilical to provide a view of the surgical site. The surgeon, however, loses depth perception and is limited to a single viewpoint. With support from the system described above the surgeon's visual interface is significantly enhanced, and the use of the network creates many additional possibilities. The video can be distributed to remote students for training purposes. A subspecialist in a different city can assist a less experienced surgeon, and even telesurgical procedures can be performed in distant or dangerous environments, such as the battlefield or a space station.

Telesurgery is an example of a general category of applications known as *teleoperation*. These applications are characterized by the need to perform some physical task in an environment which is inaccessible. Typically, a robotic device situated at the work site is controlled by one or more remote human operators. One such system is designed for carrying out tasks in a nuclear reactor facility, where the danger from radiation precludes the possibility of humans performing the required duties in safety [26]. Another system is designed to automate the dangerous and labor-intensive mining industry [66], and a system for handling hazardous materials [25], such as bomb disposal, is being developed. For these types of tasks, it has been shown that stereo vision is essential [17]. Thus, a

multi-viewpoint audio and video system not only provides ideal support for teleoperation applications by delivering stereo video from a remote site, but by supporting multiple views it also enables several operators to control different robotic functions.

A closely related concept to teleoperation is *telepresence*. This technology is usually characterized by the use of multiple video cameras to present an observer with a stereoscopic view of a remote scene. Mechanisms are also included to allow the user to make perspective changes. To a certain degree this represents a more passive system than teleoperation, but the visual interface is more robust, providing a sense of immersion. Typical uses include telerobotic inspection of "hostile" environments such as undersea [90] or outer space [93]. Overlaying graphics onto the video enables many industrial applications such as computer aided design, mechanical device repair, and interior design [1]. A multi-viewpoint audio and video system clearly provides the necessary support for such applications. A slightly modified version of telepresence can be used in a variety of applications which require the ability to "see through" some object. For example, the operators of an armored vehicle, such as a tank, must be able to look out in all directions. Utilizing an architecture such as the one we propose, cameras can be mounted on the skin of the tank and the video transmitted to the operators [74]. Each crew member can utilize a unique view, as well as shift perspective and look out a different side of the tank.

By definition the distributed, multi-participant nature of the system outlined above enables numerous remote collaboration applications such as video conferencing, computer supported cooperative work (CSCW), and remote learning [12]. In addition virtual environments in which many geographically dispersed users may work together in a common virtual space represent an interesting class of applications that can be better supported by multi-viewpoint audio and video. Examples of such projects include SIMNET [14] and the MR Toolkit Peer Package [84].

Finally, one of the potentially largest uses of this type of technology will be in the entertainment industry. Applications ranging from broadcast 3DTV [78] to video on demand to fully immersive and interactive games which blend computer generated graphics with multi-viewpoint video will be realized [12], [44]. All the applications mentioned above break new ground and represent an exciting future for computing, and systems such as the one being proposed will play a vital role.

2 Proposed Work

Our goal is to create a system capable of supporting numerous streams of audio and video in a distributed environment with multiple participants. The system should provide for non-trivial interaction with the media, and it must address the issues outlined in Section 3 below. The architecture we propose is comprised of three types of components: servers, network managers, and clients. Broadly speaking, a server is responsible for collecting and transmitting media data; a network manager ensures efficient and effective use of network resources; and a client supports interaction with and synchronized display of media data.

A typical configuration will consist of several servers and clients communicating through network managers, as shown in Figure 2. Each server will generally have multiple clients, thus making it advantageous to utilize multicast connections within the network. Each client may also receive data from more than one server. We assume a communication model with fairly persistent connections between clients and servers[1]. A protocol such as RSVP can be utilized to establish connections and reserve resources for a communication session, as well as to add or drop parties once

1. Note that a single host may act as both a client and server.

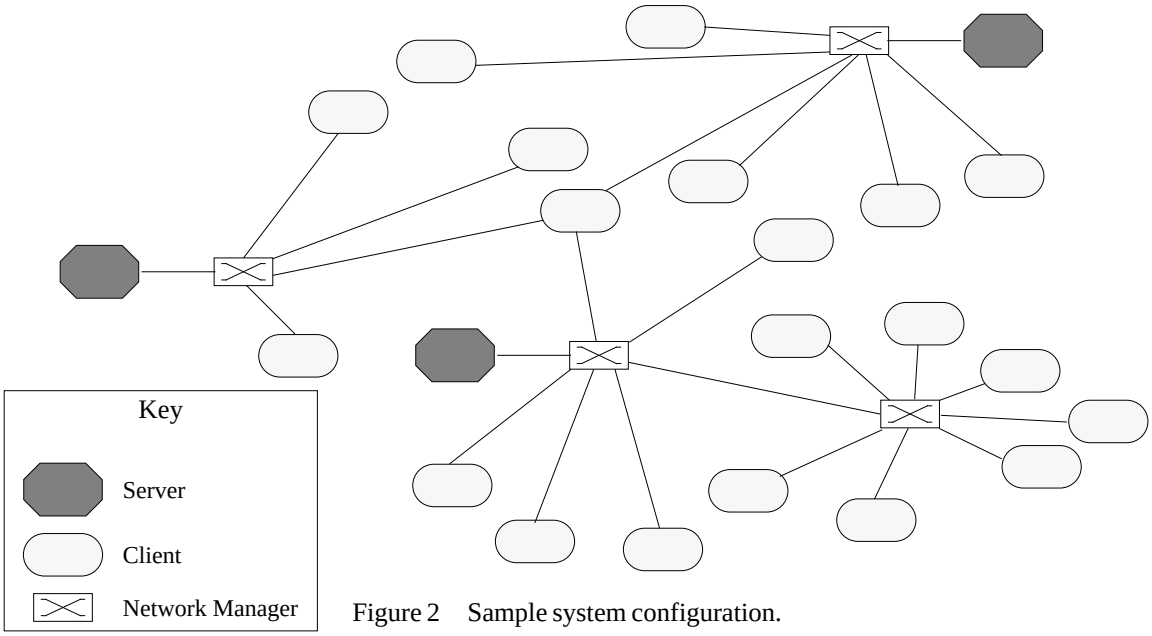

Figure 2 Sample system configuration.

a session has begun. Within the scope of these connections numerous media streams may be transmitted. Each server sends several streams along its outgoing connection, but each of its clients will typically only receive a subset of those streams. Although there will be considerable overlap between the sets of streams different clients accept, the assumed model of interaction is for the composition of those sets to change frequently and often. This requires the network to support dynamic and fine-grain modifications to multicast groups. Thus, communication in the proposed system involves both coarse-grain connection management in the form of call set-up and tear-down, as well as fine-grain connection management in the form of dynamic changes during a session.

A minimal system configuration is depicted in Figure 3. Arrows indicate the flow of data and control information; dashed arrows represent the communication of temporal information; and pipes indicate network transmission of multiple streams through a single connection. In the following sections we present a detailed description of each system component.

2.1 Server Architecture

Servers consist of camera and microphone managers, a system clock, a synchronization manager, a buffer manager, as well as disk and network interfaces (see Figure 3). The camera and microphone managers are responsible for capturing, digitizing, and optionally compressing the audio and video data from the camera and microphone arrays. They query the system clock to generate timestamps and attach them to audio and video MDUs. The system clock is used as a global time reference, and a network synchronization protocol such as NTP [57] could be used to synchronize it with other clocks on remote servers so that timestamps are meaningful across hosts. No attempt is made, however, to synchronize the system clock with remote clients. Provided that the clocks on both clients and servers are reasonable approximations of real-time, observers are extremely unlikely to notice any anomalies between the capture and display rates. The timestamped MDUs are then passed to the synchronization manager, which is primarily responsible for determining appropriate starting and stopping points for each stream so that inter-stream synchronization can be achieved at the display site. If related streams are captured by multiple servers, this process may involve out-of-band communication between servers to negotiate the positions of such synchronization points.

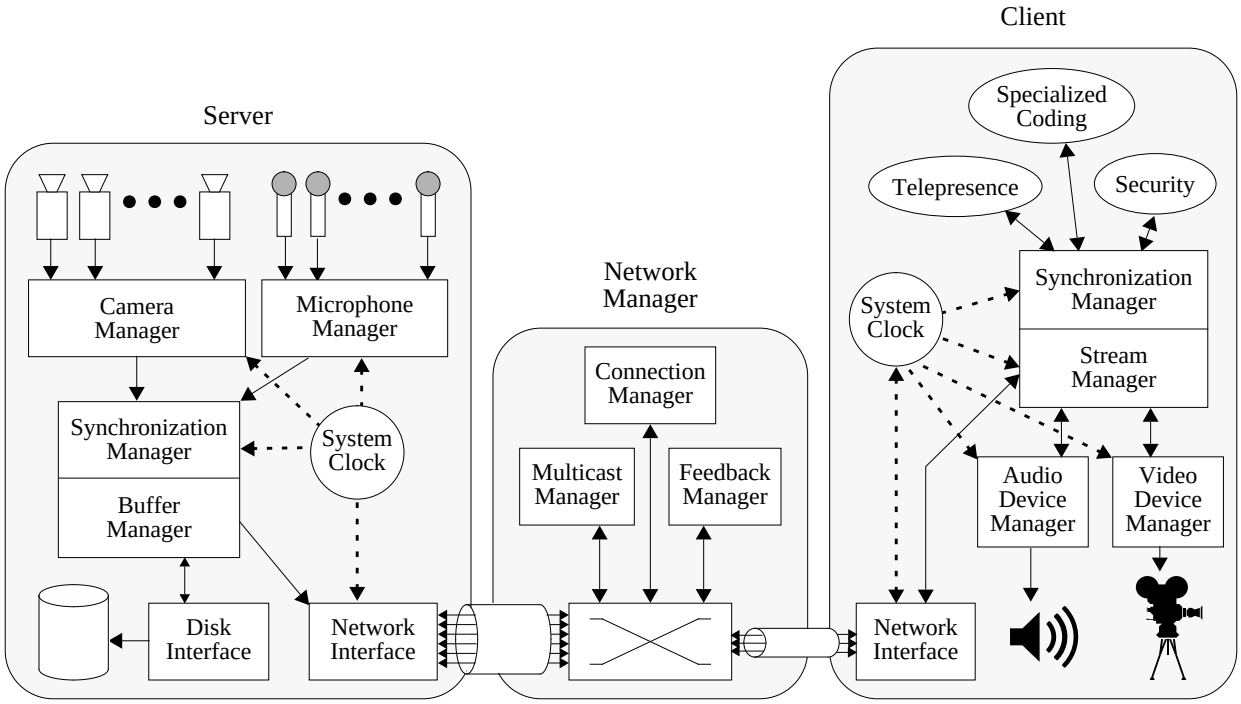

Figure 3 Architecture components.

Once this stage is complete, the MDUs are held by the buffer manager until they can be transmitted or stored. The disk interface is responsible for writing MDUs to disk using an efficient data placement strategy, as well as an appropriate disk scheduling algorithm. Data may later be retrieved by the buffer manager for subsequent transmission across the network.

The network interface has several responsibilities. It provides coarse-grain connection management, including call set-up and tear-down, resource allocation and reservation, and QoS negotiation. It must also collect MDUs from the buffer manager and organize them into separate streams. Next, MDUs from each stream are either aggregated or decomposed (depending on their size) into network protocol data units (PDUs), which are then transmitted on the outgoing connection. Finally, the network interface accepts feedback from network managers and clients. This information may be utilized to adjust transmission properties, such as negotiating a new QoS or changing the data rate.

2.2 Network Manager Architecture

Network managers reside at switches located within the network, and they consist of a connection manager, a feedback manager, and a multicast manager (see Figure 3). The connection manager is responsible for coarse-grain connection management and QoS control. Clients and servers will utilize a protocol such as RSVP to establish connections and reserve resources, and the connection manager provides the means for carrying out such requests.

The multicast manager utilizes a fast signalling protocol to provide support for fine-grain connection management. This protocol is used to perform dynamic modifications to multicast group configurations, as well as to determine the best point within the network make such changes. This

fine-grain control over multicast connections not only allows clients to quickly adjust the set of streams they receive, but it also enables multiple clients with heterogeneous capabilities to utilize a single multicast connection without restriction.

Finally, the feedback manager monitors local network state (in terms of utilization, congestion, etc.) and delivers statistics to interested clients, servers, and remote network managers. This information may be attached to regular data as it passes through the network. The feedback manager is also responsible for coordinating upstream feedback from the clients. If possible and appropriate, it will respond to the feedback messages it receives; otherwise, it passes the information further upstream. Some possible responses include instructing the connection manager to adjust its resource allocation, or requesting that the multicast manager modify which streams are destined for a particular client.

2.3 Client Architecture

Clients are also comprised of several components: applications, a synchronization manager, a system clock, a stream manager, audio and device managers, and a network interface (see Figure 3). At the highest level applications interact with the synchronization manager to control the display of media streams. Available operations include commands to open and close streams, to start and stop stream presentation, and to group streams into sets that should exhibit inter-stream synchronization. In addition cache control primitives allow applications to specify which set of open streams should be buffered by the system, thus providing fine-grain control over connection management. For example, our telepresence application will begin by opening all the media streams it may wish to have displayed and then specifies a subset to be cached. The system responds by opening connections to appropriate servers and buffering incoming data for the cached streams. Next, the streams to be viewed are started, and display begins. The group primitive is used to guarantee synchronization between audio and video, as well as between stereo video pairs. Finally, as the application runs, it will use the start and stop commands to switch streams in response to user head motion, and it will utilize the cache control commands to maintain a neighborhood of buffered streams around the user's current position so that the latency of these changes can be absorbed.

The synchronization manager provides somewhat similar functionality to the synchronization manager on the server side. It is responsible for establishing connections to servers by making appropriate requests to the network interface, and it stores stream state, such as which streams an application has opened, which streams are being cached, and which streams should be synchronized with each other. It controls when the display of streams should be started or stopped by sending the appropriate commands to the stream manager. These requests include a global time at which the actions should be performed. The system clock provides the necessary time reference; and as with servers, a network synchronization protocol, such as NTP [57], can be used to align clocks on multiple clients so that timestamps are meaningful across hosts. If inter-stream synchronization must be performed between multiple clients, the synchronization manager is responsible for sending stream group information to the appropriate remote sites. In addition negotiation with remote clients may be necessary to agree on the start and stop points for such streams. To reduce the latency of these negotiations out-of-band communication can be used.

The stream manager buffers all cached streams in order to absorb any transient timing anomalies introduced by the system. MDUs are received from the network interface and temporarily stored in appropriate buffers. A timestamp is affixed to each such MDU. This timestamp represents a future value of the system clock and indicates the time at which the MDU should be displayed. A subset of

the cached streams will be displayed at any given time, and the MDUs for each of these streams are stored only until they can be delivered to the appropriate device managers. MDUs for other streams are discarded once their presentation times have passed. The stream manager also monitors its buffers and sends it is responsible for changing the composition of the set of cached streams by sending appropriate instructions to the network interface.

The audio and video device managers receive MDUs from the stream manager and buffer them until their presentation time is reached. It may skip or duplicate MDUs or take some other corrective action in order to synchronize the device clock with the system clock. This helps ensure that presentations do not fall out of synchronization due to differing clock rates at the devices. Device managers generate exceptions when deadlines are missed or data is unavailable at the necessary time. This allows applications to take domain-specific corrective actions. Finally, the presentation latency of individual MDUs (including decompression time) is carefully monitored by the device managers so that display can be started enough in advance to ensure it completes at the appropriate time.

The network interface provides several services to other system components. It supports coarse-grain connection management and QoS control, which is utilized by the synchronization manager to establish communication with servers and other clients. The out-of-band negotiation protocol described above is also the responsibility of the network interface. A fast signalling protocol is executed by the network manager in response to requests from the stream manager for changes in the set of received streams. This is the client-side counterpart to the protocol used by multicast managers and is used to instruct network managers to make dynamic modifications to multicast group configurations. The network interface also coordinates feedback data. It monitors local network conditions and receives status messages from upstream sites. This information is passed to the synchronization and stream managers, which may respond by renegotiating QoS parameters, adjusting the requested data rate, etc. In addition lightweight feedback messages indicating the status of the client is passed upstream to interested network managers and servers. Finally, the network interface is responsible for receiving media data in the form of network PDUs, de-packetizing them into MDUs, and passing the MDUs to the stream manager.

3 Key Issues

Designing and implementing the proposed architecture will require many technical challenges to be overcome, and we outline a few of the key difficulties in this section. Although we cannot address every issue within the scope of this work, it is important to obtain a sense of the problem size.

The architecture described above is quite novel. As such, there is an inherent level difficulty associated with assembling the various parts into a functional system. Various trade-offs and compromises will be required so that different pieces of existing technology can be artfully combined to create a new and advanced multimedia environment. Building such a system involves numerous problems hidden within intricacies of the details, and flushing out those obstacles represents a significant challenge.

Since the proposed system must support numerous streams of audio and video, substantial bandwidth will be required to memory, I/O devices and the network so that media acquisition, processing, transmission and display can be performed efficiently. Designing cost-effective, high-bandwidth machine architectures is a significant problem that has not received adequate attention in the past. Research efforts have been primarily focussed on bandwidth reduction. However, the use of many media streams demands even cheaper and more efficient compression techniques. Still, as more bandwidth is made available (through new architectures or compression

schemes), it will be consumed by applications with increasing demands, and bandwidth problems will continue to exist. Thus, lack of sufficient resources will be the dominant condition, and appropriate means for dealing with such a situation must be developed.

Utilizing the network adds significant complexity to a multimedia system. Efficient packetization of media data and stream prioritization are important problems that must be addressed. In addition integrating time-dependent media data into a standard network will require a variety of service policies to be provided, ranging from best-effort to guaranteed performance. Since resource reservation is a necessary component of such an integrated network, the difficulties associated with connection and resource management increase dramatically. Further, renegotiating QoS parameters once a connection has been established becomes a significant issue. New protocols and hardware support are required to address these needs.

Providing dynamic, fine-grain control over multicast connections (as described above) poses several difficult problems. A single site will multicast a large set of streams to a group of receivers, but each receiver will accept a different subset of the streams. In addition the set of streams a receiver wishes to receive, will change frequently and on extremely short notice. This requires a fast signalling mechanism to make the appropriate multicast group modifications in a timely fashion. A protocol to determine the best point within the network to make such changes will also be required. Finally, feedback from the receivers in the form of rate and QoS control must be coordinated so that meaningful, receiver-specific service policies can be supported.

The proposed architecture will also require operating system support for local resource management. One of the main concerns is adequate scheduling support. Time-critical activities must be integrated with traditional computational entities. In addition time-driven resource management must be utilized for devices to ensure timeliness constraints are met. Since overload and error conditions are inevitable, the system must degrade service in a graceful fashion.

Media synchronization refers to the problem of aligning media acquisition and presentation activities relative to a physical clock, and it is regarded as unique and fundamental to the area of multimedia. There are many facets to this problem. Related media streams that captured by different hosts must be properly synchronized so that presentation at the display site can be performed correctly. This involves determining the capture rate for each stream as well as their relative starting points. To smooth transient timing variations within the system, the display site must employ a buffering scheme, and buffer status must be monitored so that adequate rate control between a remote transmitter and the display site can be achieved. This may involve the development of some type of software phase-locked loop. The buffers at the display site can also be used to trade switching latency for bandwidth. Since switching between streams involves costly network operations, it is beneficial to buffer a neighborhood of streams at the display site so that most transitions can occur locally. Sites which are close to the transmitter may buffer only a few streams, while more distant sites may need to buffer several streams to adequately hide switching latency. Synchronized presentation of media data requires that appropriate temporal control be placed as close to the devices as possible in order to display streams at the appropriate time and rate. Finally, appropriate abstractions should be provided so that applications other than multi-viewpoint audio and video can make effective use of the proposed system.

In addition to the issues outlined above there are a few application-level problems which must also be addressed. An appropriate camera configuration which captures a wide range of user head motion as well as a coding scheme which exploits the spatial redundancy between cameras must be developed. Head tracking will be used at the display sites to select streams for viewing. Hence, to

ameliorate the effects of latency from the tracking device, it is important to develop an accurate algorithm for head motion prediction. Switching between streams is an inherently discontinuous process, and appropriate switching points must be chosen so that no disruptions are noticed by the user. Since humans are more sensitive to sound anomalies than to video, this is of particular concern for audio streams.

4 Related Work

One of the noteworthy features of the proposed architecture is its multidisciplinary nature. Several technologies must be successfully combined to create a working system. In this section we review some of the research results which are relevant to the design and implementation of such a system.

4.1 Architectures for Multi-viewpoint Video

Several systems have been designed to support multi-viewpoint video in some way. Hence, they incorporate various features of the proposed scheme. In this subsection we examine a few of these architectures.

Fisher constructed an interactive, viewpoint dependent system for the stereoscopic display of static images [30]. A single camera was used to record snapshots of natural scenes. The camera was translated horizontally, vertically, and in depth. This "volume" of images was then digitized and laid out on video disc. An observer could then view the scene from numerous perspectives, and a body tracker was used to dynamically determine the correct frames to display. Stereopsis was achieved through the use of stereo shutter glasses and field sequential display of laterally displaced views. Such glasses are synchronized with the monitor so that when the image for one eye is presented, the glasses darken for the other eye. In this way stereo pairs are alternately displayed in each eye. Since this system is only capable of playing recorded views of a static scene, its usefulness is limited. This is perhaps due to the fact that it predates modern network and compression technology. However, it forms some of the basis for the proposed work.

The Virtual Dome [38] is based on the same principles as Fisher's system, but the scene is not recorded and may be dynamic. A single video camera is mounted on a rotating platform which pans and tilts through a semi-spherical region. As the camera moves, it takes snapshots of this region from 75 contiguous perspectives. The analog video signal is then transmitted along a direct connection to a host machine for display. The signal is digitized, and images from each of the perspectives are texture mapped onto a spherical object, the Virtual Dome, which is viewed through a head-mounted display (HMD). Head position is monitored so that the correct perspective is observed. Ideally, the Virtual Dome would be continuously updated, but interactive requirements prohibit the lengthy refresh rate. To compensate for this problem, the system performs conditional replenishment, updating only the neighborhood of the currently viewed perspective. The Virtual Dome supports a single observer, and the use of a direct connection to transmit analog video does not scale. Hence, this system is only useful for a limited class of applications.

The VIEW project at the NASA Ames Research Center is aimed at providing head-coupled stereoscopic viewing for telepresence applications [8]. This system utilizes two black-and-white NTSC video cameras mounted on a computer controlled platform. The cameras are arranged so that: 1) the optical axes are parallel, 2) the image areas are coplanar, and 3) the bottom edges of the image areas are colinear. This ensures that the view of one camera is a simple lateral displacement from the other camera and is referred to as the *parallel axes configuration* (see Figure 4). The analog video signals are directly connected to an HMD. Head position is monitored by a host computer in real time,

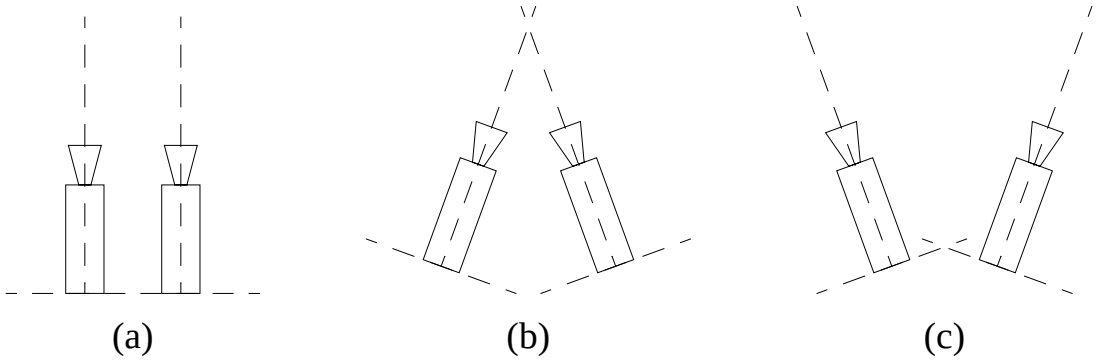

Figure 4 Camera configurations: a) parallel axes, b) verged, c) divergent

and tracking information is used to adjust the pan, tilt, and roll of the camera platform. This enables the observer to view a remote scene from any stationary perspective. Although this provides a wide field of view, the ability to look around objects can only be supported by a mobile platform. By its nature this system is restricted to a single user, and the method of interaction limits the useful distance between the cameras and the observer. Using a general purpose network to transport the video would increase system flexibility, but the cost in terms of added complexity would be high.

A closely related system that was also developed at NASA Ames is a telepresence controlled remote operated vehicle (TROV) for underwater exploration [90]. A similar camera arrangement and platform are used, but the video signals are digitized, compressed and transmitted across a satellite link to a remote workstation. Stereo pairs are displayed field sequentially, and stereopsis is achieved with stereo shutter glasses or an HMD. This system differs from the proposed architecture in that a dedicated network is used, and a single user is supported. Further, viewpoint changes to look around an object can only be effected by TROV motion.

Look around is the ability to shift the viewpoint laterally so that previously occluded objects can be observed. The NASA TROV offers only limited look around functionality, but two closely related systems have extended the pan, tilt, roll model of camera motion to include horizontal and vertical movement, thus providing full support for this highly desirable capability. The system in [34] utilizes a single camera with the front and back mounted on separate plotter arms. By controlling the plotters, the camera can be moved in all directions, with the exception of rolling about the optical axis. The Remote Surface Inspection project at Jet Propulsion Laboratories mounted a pair of cameras onto the end of a seven degrees of freedom robotic arm [93]. The cameras are arranged in the parallel axes configuration, and by controlling the movements of the robotic arm, virtually any camera angle can be achieved. Both systems send the video signals directly to a host workstation for display on a monitor. The RSI system utilizes field sequential presentation of stereo pairs in coordination with shutter glasses to provide stereoscopic views. Both systems use head tracking to determine the desired perspective for observation, and the camera(s) are then moved to provide the appropriate view. Both systems are limited to a single user, and video transmission across a shared network is not supported.

The active multibaseline stereo system from Carnegie-Mellon University [31], [45], [97] utilizes four laterally displaced black-and-white NTSC video cameras. The cameras are verged so that their optical axes intersect at a common point, and they are vertically aligned so that the bottom edges of their image areas are coplanar (see Figure 4). This configuration increases the volume defined by the intersection of the fields of view from the cameras. Such an arrangement is useful for robotic vision

applications. A specialized A/D converter samples and digitizes each of the video signals. An 8-bit luminance value is produced for each sample, and corresponding samples from each stream are concatenated into a 32-bit word and input into an iWarp parallel machine. The raw video data is then utilized in computationally intensive vision applications. Since the intended observer is a robot, this system does not share the same design goals as the proposed architecture. It does not distribute the video to multiple observers across a network, and there is no means for interaction. Since precise data is required to accurately recover depth information, no compression is performed. This requires more bandwidth and thus limits the number of views which may be utilized.

Holocam from Sun Microsystems is perhaps the most closely related system to the proposed architecture. Holocam utilizes an array of thirty-two full-color NTSC video cameras arranged in the parallel axes configuration. The cameras are genlocked, and the signals are routed through a 32x2 crossbar switch. The two video signals output from the switch are connected to a standard video capture card in a host workstation. The signals are digitized and displayed field sequentially. Stereopsis is achieved through the use of LC shutter glasses. Head tracking functionality is built into the glasses, and the wearer notifies the system of desired viewpoint changes through lateral head motion. The system uses velocity and acceleration information to predict head position at the display time for the next set of frames, and then instructs the crossbar to output the appropriate signals. In this way look around is supported without the need for camera movement. No compression of the video data is attempted. Holocam does not utilize a network and is thus limited to a single observer. Since only two video signals are captured at a given time, dynamic viewpoint changes are restricted to live input.

A common feature of all the systems described above is the limit to a single user. This is not a weakness inherent in the design of these architectures, for the video signals can all be replicated and delivered to multiple destinations. However, this forces all observers to view the scene from a predetermined perspective. If user control over viewpoint changes is to be supported, some mechanism for arbitration would be required. A system which allows individual control over perspective changes would be much more advantageous. Another limitation of the above systems is the use of dedicated transmission lines to carry the video to the display site. Utilization of a shared, general purpose network to distribute the video would not only allow multiple users to receive the video and interact, but it would also dramatically increase the flexibility of the overall system.

4.2 Multi-Viewpoint Video Coding

The proposed architecture will utilize numerous cameras to acquire video for transmission across a network. This represents a significant expense in terms of bandwidth, and thus, some means of compressing the data is required. Since our representative application, multi-viewpoint audio and video, exhibits a high degree of spatial redundancy between streams, it is important to investigate coding schemes which exploit that property. In this section we examine the human factors related to the compression of stereo video, as well as some proposed coding schemes.

4.2.1 Human Factors

Several studies have been conducted to observe the effects of lossy compression on the perceived quality of a stereo pair. Harrison and McAllister [37] report that in general it is important to retain high fidelity in compressed stereo images so that human observers can achieve comfortable fusion. Some lossy reproductions of are easily detected as being worse than the originals, while others appear

identical at first but lead to eye strain and discomfort after prolonged viewing. Viewer illness is an issue which is not a consideration for coding of 2D images but which must clearly be taken into account for 3D compression.

Kost and Pastoor [47] considered the problem of quantizing disparity information for stereo images. Disparity refers to the parallax effect between two views of the same scene. A point in the scene will generally be perceived in different a location for each image, and the difference between those locations is a measure of *disparity*. Many proposed coding schemes for multi-viewpoint images suggest the use of a single image coupled with disparity information to reconstruct the other views. Compression is achieved by quantizing the disparity data. Reconstruction of the stereo pair can result in most or all of the quantization errors appearing in one image. Psycho-optical experiments demonstrated, however, that distributing these errors equally between the images produced results of subjectively higher value. In related experiments Pastoor and Beldie [69] demonstrated an inverse relation between correlated noise patterns and subjective quality. Higher correlation in errors between images of a stereo pair result in diminished quality. These results suggest that compression errors should be randomly distributed between images pairs in such a way that each image receives an equal portion.

Closely related to the issue of disparity quantization is the threshold for temporal changes in disparity to be perceived as movements in depth. The upper bound lies in the range 4–6Hz [68]. This suggests that disparity differences can be quantized temporally between successive frames of a 3D video sequence. However, sudden changes in disparity result in discontinuous motion of objects in depth, which is more visually tiring than smooth motion. These factors should be considered when devising an appropriate multi-viewpoint video coding scheme.

Contrast sensitivity is higher for binocular versus monocular viewing, and a binocular-monocular sensitivity ratio of 1.4 has been found in many studies. Half-occluded regions are those areas of a stereo image that are visible to one eye but not the other. Since binocular acuity is higher than monocular acuity, it should be possible to compress high spatial frequencies in these monocular regions without decreasing subjective quality. However, Liu [54] demonstrated that impairment visible in binocular regions was also visible in half-occluded sections. Thus, non-uniform compression of stereo relevant portions of an image pair does not appear to be advantageous.

Perkins [71] compared two techniques for compressing stereo pairs: disparity-compensated transform domain predictive (DCTDP) coding, and mixed-resolution coding. DCTDP coding is representative of a large class of compression methods. These techniques typically consist of the following step. One image, say the left, is coded independently with any desired scheme and transmitted. Next, the right image is decomposed into rectangular blocks, and for each of these blocks the encoded left image is searched for a block which most closely approximates it. The locations of these approximation blocks is then transmitted as side information. Finally, the transform of the right block is estimated from the transform of its approximation block, and the difference between the original and predicted blocks is computed. This error signal is quantized and transmitted. Such a scheme can produce high degrees of compression, but the computational expense can be quite large. Thus, Perkins suggests the use of mixed-resolution coding, i.e. one image is coded with high fidelity at full resolution while the other is low pass filtered and subsampled. This method takes advantage of the concept of *binocular rivalry*, which states that if two images do not have identical frequency components, the one that contains the high frequencies will dominate the perceived view. Thus, when a filtered and unfiltered image are fused, the final perception will be much closer to the unfiltered

image. There is a cost, however. The ability to distinguish fine depth differences is slightly decreased. Still, users reported comfortable viewing and higher subjective quality over DCTDP schemes for stereo pairs consisting of an original image and one decimated by four both horizontally and vertically. Combining these two techniques into a hybrid approach can yield good compression with high perceived quality.

Finally, Pastoor and Schenke [70] conducted experiments unique to multi-viewpoint systems. Horizontal translation while viewing a fixed stereo pair results in an unnatural rotation of the image. Objects in front of the screen move in the same direction as the viewer, whereas objects behind the screen move in the opposite direction. Multi-viewpoint systems help limit this rotation effect by providing new views of the scene. However, view changes occur discretely and discontinuously. When a perspective change occurs, there is a discrete change in parallax, which produces an effect known as *flipping*. User studies showed that the rotation effect was considered less annoying than flipping, and to achieve good reproduction of multi-viewpoint images so that flipping was considered perceptible but not annoying, at least nine views per inter-ocular distance are required. This result assumes a maximum disparity of ten minutes of arc and represents a stringent requirement. Cinematography, however, may include images with disparities up to seventy minutes of arc (the limit for comfortable fusion), which will require sixty views per inter-ocular distance. Clearly, this necessitates a camera density which is much higher than is practical, and thus some means of interpolating intermediate viewpoints is needed.

4.2.2 Coding Schemes

While coding of 3D images has not received the widespread attention in the literature that 2D compression has enjoyed, several different stereo coding methods have been proposed. In this section we review some of these schemes.

Horst [40] proposed a simple DCTDP scheme for coding stereo video sequences. The left stream is compressed independently using the MPEG algorithm. The right stream is coded in a similar manner, but disparity estimation is used to complement temporal motion prediction. Disparity estimation is implemented as an extension of motion estimation by searching for the block in the left image that has minimum mean squared error. Typical disparity values are in the range 63 pixels horizontally and 3 pixels vertically, and so the search window is constrained to that region. This method is easily implemented for stereo sequences, but the cost of extending it to support numerous video streams exceeds its usefulness. In addition, as mentioned above, a mixed resolution compression algorithm is less computationally challenging and achieves better results.

Dinstein et. al. [24] implemented a mixed resolution scheme for still stereo images. In their experiments they evaluated two versions of the coder. In the first approach one image was left in its original form, and the other was low pass filtered and subsampled by means of a Gaussian pyramid. Compression is achieved by selecting one of the levels from the pyramid to use in place of the second image. The stereo pair may be viewed by expanding the subsampled image to full size. The second method extended this approach by performing a DCT compression step on the subsampled and full size images. User studies demonstrated good results and accurate depth perception for both versions when the subsampled image represents a reduction by two both horizontally and vertically. However, slight depth inaccuracies were reported for subsampling by four in both dimensions.

Researchers at Columbia University [94] have investigated means for incorporating stereo video compression into the MPEG II coding standard. The main concept is to encode one channel in the base layer, perform a prediction operation to estimate the other channel, and transmit prediction error

information in the enhancement layer. Since MPEG II allows multiple enhancement layers, this method applies equally well to multi-viewpoint sequences as well. Several schemes are proposed, each utilizing different combinations of motion and disparity estimation to form the predictions. In all cases it was found that better SNR ratios were achieved over the case of coding each channel independently. While this scheme has the benefit of compatibility with an established standard, the cost of implementation in terms of design effort and required computational resources is extremely high. In particular, when numerous video streams are to be coded, as in the proposed architecture, the engineering effort alone would be excessive.

Gunatilake et. al. [36] propose a scheme which essentially extends the above approach to take advantage of the binocular rivalry effect. A single frame is coded in one of four ways. Intraframe coding performs 2D compression to remove spatial redundancy using an algorithm similar to the JPEG standard. These frames correspond to I-frames used in MPEG. M-frames are coded based on forward motion estimation and error compensation within a single channel, similar to MPEG P-frames. Similar to MPEG B-frames, bidirectional motion estimation is used to interpolate B-frames within a single channel. Finally, a "worldline frame," or W-frame, is composed of the best estimation blocks chosen from a frame in the recent past within the same channel, as well as the three most recent frames in the other channel. Two video channels are coded with a sequence of these frame types in such a way that a high resolution frame is always paired with a low resolution. This enables the algorithm to achieve greater compression than an MPEG scheme for the same level of quality. The major drawback of this method, however, is that it was designed for stereo compression only and does not scale to larger numbers of video streams.

One of the most common methods of computing disparity between two images is to decompose one image into fixed size blocks and then search for the closest approximation to each of them in the other image. These disparity vectors can be used to predict one of the images. A significant problem with this approach are distracting edge artifacts which result from the use of fixed block sizes. Sethuraman et. al. [83] seek to alleviate this problem by partitioning an image into variable size blocks of roughly uniform disparity. This segmentation is based on a quadtree method which divides blocks based on their disparity compensated error. The result is disparity predictions which are more efficiently represented and more accurate than for fixed block schemes.

Fujii and Harashima [33] describe an approach to compress multi-viewpoint video sequences by extracting structure information from the scene. A great deal of work has been done in the area of machine vision to analyze methods for recovering depth from stereo video. Using these techniques, depth information can be recovered and used to create a 3D wireframe model of the scene. The video data can then be used to create a texture map for the scene. The algorithm for creating this 3D model is somewhat simplistic and does not adequately handle fine textures and occluded regions. Despite these shortcomings, there are several advantages to this approach. An effective compression ratio of 1/N, where N is the number of video streams, is achieved. Integration of 3D graphics into the video is facilitated by the use of a graphical model. Finally, interpolation of intermediate viewpoints becomes a simple perspective change, a common operation for graphics applications. Although these benefits make this scheme quite attractive, there is a major stumbling block which make it undesirable. The computational complexity makes the problem intractable with current hardware. Several minutes were required to encode sequences which utilized nineteen viewpoints. Even with advances in processing power, this approach will not be feasible in the near future.

As mentioned above the required camera density for multi-viewpoint video can be quite high. To create a practical system a smaller number of cameras will be utilized, and intermediate views will be synthesized through an interpolation process. This can be accomplished by a simple scaled translation of corresponding pixels from reference cameras, and constructing expensive 3D models is not required. The hard part, however, is determining which pixels in a pair of images correspond to the same point in the viewed scene. This is known as the *correspondence problem*. One solution is to perform the equivalent of block-based motion estimation between the two reference images to produce a set of disparity vectors. Although this approach is somewhat naive, it can yield good quality results. To achieve better accuracy, Liu and Skerjanc [55] suggest the use of three cameras arranged in a triangular fashion. The nature of the camera geometries enables disparity estimates made between two images to be verified by the third. Experimental results showed good performance on still images, but the correspondence algorithm is computationally intensive and could not execute in real time. Katayama et. al. [46] suggest another approach which utilizes an array of camera arranged in the parallel axes configuration. Corresponding scan lines from the different cameras are arranged to form a 2D image, i.e. a scan line from the first image forms the first row, a scan line from the second image forms the second row, etc. Due to the camera geometry corresponding points will lie along straight lines within these images. An algorithm is presented which identifies such lines in an iterative fashion, and good results are achieved for images interpolated both between and in front of reference cameras. Once again, however, the computational complexity is prohibitively expensive for video sequences to be interpolated in real time.

4.3 Networking

The network plays a major role in the proposed architecture. Many issues must be resolved, ranging from appropriate QoS support to fine-grain connection management. In this section we review some of the work that has been done to investigate these topics.

4.3.1 Coarse-Grain Connection Management

In order to establish communication between clients and servers (see Section 2) a protocol for coarse-grain connection management is required. Since media streams possess timeliness constraints, certain guarantees regarding service quality may be required from the network. One method of ensuring QoS parameters can be met is to reserve resources within the network. RSVP ([101], [102]) is one such resource reservation protocol. It operates within network routers above the IP layer. Standard techniques are used to establish unicast or multicast connections, which can be utilized in any combination to establish multi-party communication. Receivers initiate reservations by transmitting resource requests upstream toward the sender(s). Each router that receives such a request makes a local admission control decision. If there are sufficient resources, the request is granted, and resources are reserved on the downstream link toward the receiver. Otherwise, an error message is sent to the requestor. Reservations take the form of *flowspecs*, which are not defined by RSVP and left to the router and application to provide. Requests from multiple receivers for the same set of senders are merged if possible so that a single reservation is used to accommodate the data flows. Soft state is maintained by the routers to record the reservations, as well as backward paths from receivers to senders. Periodically, senders transmit *path messages* downstream, which are used to update the backward paths so that dynamic routing changes can be accommodated. Similarly, receivers send *reservation messages* upstream to refresh the reservation state. If no such refresh messages are received over a fixed interval, the routers discard the state, and reservations must be

re-established. In this way RSVP provides a method for reserving resources within network routers for arbitrary and heterogeneous sets of senders and receivers, but the policies and mechanisms to establish and enforce these reservations must be provided by other means.

The Internet Stream Protocol, Version 2 (ST-II [22]) is another resource reservation protocol. In contrast to RSVP, ST-II is connection-oriented, and reservations are sender initiated. Although it operates at the same level as IP, ST-II is not intended as a replacement, but rather as an adjunct. Connections may be either unicast or multicast, and receivers may be added or dropped once a session has begun. *Flowspecs*, which indicate the required service quality and priority level of the stream, are utilized during this call set-up phase to reserve resources along the communication path. At each hop (host or router) a routing function is called to determine the best path to follow, and the flowspec is then given to a *local resource manager*, which is responsible for evaluating the request and reserving appropriate resources. The routing function differs from the IP routing function in that it not only seeks to minimize hops, but also takes into account available bandwidth, path delays, etc. Once a connection has been established, routes cannot be modified, but a change in the flowspec may be requested. To improve resource utilization connections can be grouped to share bandwidth, routes, subnet resources, and configuration changes. In addition to resource reservation ST-II provides a simple data transfer protocol which utilizes the established connections to transmit data with the specified QoS. Actual resource management is not specified by ST-II, however, and it is assumed that local resource managers provide all the necessary reservation and enforcement facilities. Although arbitrary combinations of senders and receivers can be supported by ST-II, each connection must have uniform QoS, which implies multicast to a heterogeneous group of receivers is not supported.

The Tenet Protocol Suite 2 also provides resource reservation as part of channel establishment ([6], [29]). Like ST-II, it is connection-oriented, but connections and reservations may be initiated by either a sender or receiver. Whereas RSVP utilizes a single pass from receiver to sender to allocate bandwidth, Suite 2 uses a two-pass approach in which resources are tentatively reserved on the initial pass (in case a later node rejects the connection) and confirmed on the return/confirmation pass. Unlike the “soft state” RSVP maintains, Suite 2 utilizes persistent objects within network nodes (e.g. routers) to record state regarding resource allocation and connection characteristics. This is possible since the connection-oriented nature of the protocol implies that routes do not change once a connection has been established. Suite 2 also includes its own routing and transport protocols, which are designed to provide certain QoS guarantees (see Section 4.3.2). Network resources may also be divided into disjoint partitions so that each one is isolated from the effects of traffic in other partitions. Arbitrary combinations of (possibly heterogeneous) senders and receivers, as well as dynamic changes to connection configurations are supported. Although Suite 2 is quite robust, channel set-up is expensive. On a Sun SparcStation 1 and an ethernet network with four hops, the average time to establish one channel was measured to be 114ms. That is too costly for the dynamic and fine-grain connection management required by the proposed architecture.

4.3.2 Transport

Multimedia applications also require an effective transport mechanism for time-dependent media data. The Real-time Transport Protocol (RTP [82]) was designed to meets this need. It does not ensure timely data delivery, relying upon lower layer services to make such QoS guarantees. Instead, RTP provides mechanisms for including timestamps, identifying payload types, sequence numbering, and monitoring data delivery. It is typically built on top of UDP, but other underlying network protocols

may also be used. RTP primarily defines the syntax of data transmission, and *profiles* are used to describe the semantics, i.e. how to interpret the fields of packet headers. Hence, RTP services are not fully specified so that individual applications can tailor the protocol to suit their needs. For example, the audio and video profile [81] lists several values for the payload type field which indicate various coding schemes, outlines the use of timestamps and sequence numbers, specifies the proper insertion of markers to indicate noteworthy points within a stream, etc. A control protocol (RTCP) is also specified. RTCP is used by applications to monitor the quality of service the network is delivering, as well as to disseminate information regarding the participants in a communication session. Thus, RTP is a useful protocol for applications to transmit media data in a domain-specific and format-independent manner, as well as to monitor and react to changing network conditions.

The Tenet protocol (see Section 4.3.1) is designed to provide QoS guarantees for network traffic in terms of minimum and maximum tolerable latency, as well as delay jitter [28]. In order to guarantee end-to-end delay bounds certain conditions must be satisfied. First, deterministic delay bounds on paths between gateways[2] are known or can be obtained. Next, gateways must utilize store-and-forward. Finally, each host system and output link from a gateway must schedule packets for transmission in such a way that it is possible to guarantee the end-to-end delay bounds specified by the client, e.g. using round robin, fair queueing, hierarchical round robin, earliest due date, first come first served, etc. All nodes along a transmission path must compute local delay and jitter bounds which ensure end-to-end bounds can be satisfied. Then, each packet that passes through the node is stored until its maximum delay time is reached, at which time it is forwarded to the next node. This scheme requires synchronized clocks between nodes, but small divergences can be tolerated. In addition to providing performance guarantees, this approach reduces buffer space requirements by spreading the buffering out evenly over the nodes in a communication path. This type of service is useful to ensure the timely delivery of media data.

The Multiflow Conversation Protocol (MCP [99]) does not provide mechanisms for making QoS guarantees. Instead, it is designed to provide correct causal and temporal ordering of messages containing media data transmitted during a multi-party communication session. For example, if participant A receives a message from participant B before it sends out its own message, all other participants should also receive B's message prior to A's. MCP ensures this type of message order by delaying the arrival of A's message until after B's. However, if B's message is lost or delayed, A's message will not be discarded, but will be delivered to other participants after a predetermined time limit has expired in order to avoid rendering A's time-critical data worthless. This is termed Δ-*causality*, which is extremely useful for cooperative working environments.

4.3.3 Fine-Grain Connection Management

In addition to the coarse-grain connection management tasks (e.g. resource reservation and QoS control) that are
provided by the above protocols, many multimedia applications require more fine-grain control. For example, the RED-VBR scheme [100] to support variable bit rate (VBR) video requires the ability to adjust QoS parameters of a delay-sensitive connection frequently during the course of a communication session. By tuning the set of resources consumed by the connection over a small time interval, network utilization can be significantly improved. The proposed architecture also requires fine-grain control over network resources but in terms of stream selection. In either case a fast

2. Computer systems connecting two distinct networks.

signalling mechanism is required to make the appropriate modifications in a timely fashion. To achieve this goal the Dynamic Connection Management (DCM [67]) scheme was developed as an extension to the Tenet protocol suite (see Section 4.3.2). DCM provides mechanisms whereby an application may alter the set of network resources it has been assigned. This may involve establishing a separate connection with the new parameters and then dismantling the original, or a supplementary connection may be set up containing only the additional resources that are required, which is then merged with the original. Bounds may be established on the amount of disruption (e.g. packet loss) that is allowed to occur during transition. Simulation studies confirmed the validity of the approach and demonstrated that modification times were short (around 120ms for a network of eleven nodes with a 2.5ms constant delay between them) and dominated by propagation delay. The DCM scheme includes a great deal of functionality, but the proposed architecture requires only the ability to switch the data source for a connection. Constraining the problem space in this manner can provide the opportunity for improved performance.

The Heidelberg Transport System (HeiTS) is a real-time communication mechanism that utilizes ST-II (see Section 4.3.2). This system includes mechanisms to provide guaranteed service and rate control on unicast connections, but the feature most relevant to the proposed work is its *discrete scaling technique* [23]. Since HeiTS supports multicast connections, it is desirable to provide a means for individual receivers to control the QoS of the delivered data. To accomplish this goal they suggest decomposing a single media stream into several connections according to importance. Then, each receiver can choose which set of connections to enable. For example, an MPEG stream could be multicast on three connections: one each for the I-frames, P-frames, and B-frames. Receivers can scale the amount of bandwidth required by simply selecting which of the connections to accept. This is precisely the type of control over connection management required by the proposed system. In HeiTS, however, connection modifications are assumed to be relatively infrequent, whereas in the proposed architecture more dynamic changes will be required. More work is needed to provide the necessary functionality.

4.3.4 Multicast

The Multicast Backbone (MBone [56]) represents one of the first large-scale uses of multicast technology to distribute media data. It is a virtual network that shares the same physical layer as the Internet, and it utilizes a collection of specialized routers (mrouters) that implement IP multicast [21] and RTP (see Section 4.3.2). The MBone is typically used for teleconferencing applications, such as "broadcasting" audio and video streams from a lecture to a group of participants. Connection management is extremely coarse-grain. An event to be transmitted across the MBone is announced in advance via an electronic mailing list. Interested parties may then join the appropriate multicast host group in order to receive the transmission. This is accomplished by sending a membership request to the nearest mrouter, which then forwards the necessary information to other mrouters in the MBone network. The mrouters are responsible for maintaining state regarding group membership and delivering the data accordingly. Hence, connection management is essentially limited to joining and leaving a session, and the cost of these operations is quite high. Although the MBone is inadequate for our purposes, it represents a useful tool for investigating network issues related to multicast.

Coordinating feedback from the receivers in a multicast connection is an important issue that the proposed system must address. One method of dealing with this problem in a scalable manner is described in [9]. In this scheme feedback is solicited by the source at periodic intervals, termed

epochs. To avoid an implosion of replies at the source a random set of receivers will respond in a statistically controlled manner. At the beginning of an epoch each participant (including the source) generates a random number, and the epoch then proceeds with a series of feedback requests from the source, termed rounds. Each request contains the random number generated by the source as well as a current timestamp. During the first round only the receivers whose random numbers exactly match the random number from the source reply. A reply includes an indication of the network load along with the receiver's response time, which is used to determine the round-trip time. In subsequent rounds receivers use one less significant bit from the random numbers in making comparisons to decide whether or not to reply. The scheme in [9] uses sixteen-bit numbers for these comparisons. Thus, in the first round all sixteen bits are used; in the second round the low order fifteen bits are used; the low order fourteen bits are used in the third round; and so on. The sender keeps track of the maximum round trip time and ends a round if no reply is received within that interval. Based on the first round in which a reply is received, the sender can obtain an accurate estimate of the number of receivers, and a probabilistic relation exists between the percentage of receivers experiencing congestion and the number of rounds from the first response to the first response indicating congestion. A control algorithm can then use this information to adjust the transmission rate at the source.

4.4 Media Synchronization

Synchronization in multimedia systems refers to the problem of assuring the correct temporal alignment of time-critical activities relative to a physical clock. This is a fundamental issue in multimedia research, and a considerable amount of work has been devoted to it. A good overview can be found in [89], and we will draw heavily upon their philosophy. We proceed by first presenting some synchronization terminology.

4.4.1 Terminology

A *media object* is a repository for *media data*, i.e. data which are intended for human consumption, such as text, graphics, audio, and video. Media objects typically consist of a sequence data units, which we will call *media data units*, or MDUs. For example, the MDUs of a video sequence could be the individual frames. MDUs serve the same role as the logical data units (LDUs) described in [89]. We choose the term MDU over LDU because it directly conveys the relationship to media, whereas LDU is media neutral. Media objects may be either *time-dependent* or *time-independent*. The presentation of MDUs from a time-independent media object does not include a temporal component. These objects are also referred to as *discrete media*. Examples include traditional media such as text and images. A time-dependent media object possesses timeliness requirements with respect to the presentation of its MDUs, i.e. correctness is a function of time. Examples include graphical animations and audio streams. Often, the MDUs of a time-dependent media object must be presented with a fixed period, as in the case of a video clip. These objects are sometimes referred to as *continuous media*. The terms "discrete" and "continuous" are unfortunate because they are somewhat counterintuitive, and because they are used inconsistently in the literature. Therefore, we will not utilize them here. Another common but inconsistently-used term is the notion of a *media stream*. For our purposes a media stream is a time-ordered sequence of MDUs comprising a time-dependent media object. Examples include video and closed captions.

4.4.2 Synchronization Classification

Given the above definitions, we can classify the media synchronization problem along several lines. First, three types time relations can be distinguished. *Intra-object synchronization* refers to interactions between the MDUs of a single time-dependent media object. For example, displaying audio at a rate of 8KHz requires that each sample be displayed for 125 s. This is also commonly known as *intra-stream synchronization*. *Inter-object synchronization* refers to the time relations between the MDUs of different media objects. For example, a presentation sequence consisting of a musical introduction followed by the display of a title image, followed by the display of a video sequence with accompanying audio incorporates both serial and parallel relations between media objects. *Inter-stream synchronization* is used to denote the case where all relations are between time-dependent media objects, and *lip synchronization* refers to the special case of coordinating audio and video streams from a movie sequence. The final type of time relation can exist between an MDU and some external event, such as user input. We call this *event-based synchronization*.

Next, synchronization is specified and enforced in terms of MDUs from media objects, but the definition of an MDU is media- and application-specific. Often, a media object may be hierarchically decomposed into potential MDUs at several levels of granularity. For example, a closed caption media object is comprised of a sequence of captions, which are comprised of a sequence of words, which are comprised of a sequence of alphanumeric characters. Any of these levels (including the entire object) could be used to specify synchronization constraints. Thus, MDU granularity also distinguishes different classes of synchronization.

Another consideration is the presentation duration of MDUs. These durations may be fixed, as in the case of a 30Hz video object in which an MDU is composed of a single frame. Fixed durations can be altered by user input, e.g. by hitting the fast-forward button during a video presentation. Variable durations are also possible. For example, MDUs consisting of single captions from a closed caption transcript can be displayed for differing numbers of video frames. User input can also result in unpredictable durations, as in the case of a slide presentation in which an MDU consists of an individual slide that is displayed until the speaker finishes discussing it.

4.4.3 Problem Statement

Based on the above discussion we can now provide a precise statement of the problem. Media synchronization refers to the requirement that all MDUs associated with a multimedia presentation be displayed at the proper time for the proper duration. A common time reference must be used to determine display times and durations for all MDUs, but these display times and durations may not be known *a priori*. Errors should be handled gracefully. In the proposed architecture we will be concerned with intra-stream synchronization for the individual audio and video streams, lip synchronization between corresponding pairs of audio and video streams, as well as event-based synchronization associated with changing perspectives in response to user input.

4.4.4 Reference Model

Steinmetz and Nahrstedt present a reference model to help characterize the requirements of a system designed to support media synchronization [89]. This paradigm is useful for defining appropriate system structures as well as interfaces between components, and it can be used to compare different solutions. The model is depicted in Figure 5, and we outline it below.

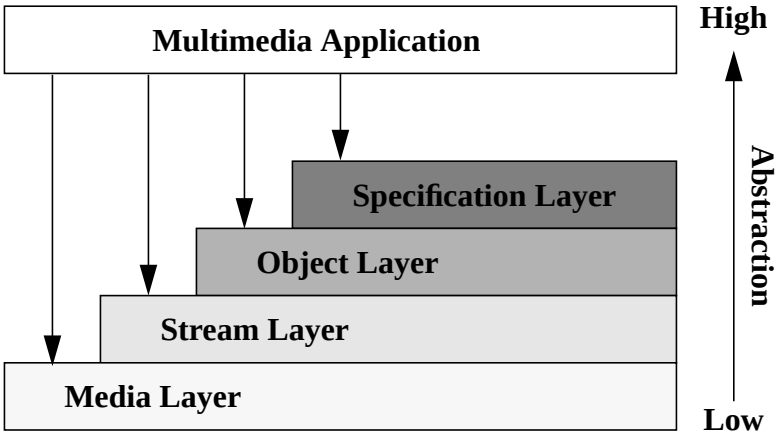

Figure 5 Media synchronization reference model.

The reference model consists of four layers which are used to specify and enforce synchronization constraints. Each layer includes an external interface that can be used to define these constraints, the level of abstraction increases with higher layers. Communication between layers occurs across the interfaces, but an application program may also utilize any interface directly. The layers are as follows: the media layer, the stream layer, the object layer, and the specification layer.

The media layer provides a device-independent I/O model for individual media streams. The interface includes operations to read and write MDUs, but synchronous operations are the responsibility of higher layers. In essence this layer provides the functionality of common device drivers, but it is important to note that only time-dependent media are considered. Time-independent media objects and user interactions are the domain of higher layers.

The stream layer is responsible for some of the most important synchronization functionality. Media streams, alone or in groups, are the units of abstraction. This layer must enforce intra-stream synchronization for individual streams, including flow control between producing and consuming devices in both stand-alone and networked environments. Inter-stream synchronization is preserved between the members of a group. Individual MDUs are not visible to higher layers. Typical operations include starting and stopping a stream or group of streams, creating a group of streams, and setting attachment points within streams. This last facility can be used by higher layers that wish to be notified when a particular point has been reached during a presentation so that time-independent objects or user interaction can be started.

The object layer provides the abstraction of uniform media objects. All types of media are supported, and the distinctions between time-dependent and time-independent objects are blurred. This layer is responsible for taking a synchronization specification and correctly scheduling a total multimedia presentation. Typical operations on media objects include prepare for display, run, stop, and destroy.

The specification layer represents the highest level of abstraction, and it is responsible for creating a synchronization specification and delivering it to the object. This may take the form of an authoring system or multimedia document editor. Several types of specifications are possible, including time flow graphs ([50], [51]), reference points within media streams [87], interval-based temporal relations in the Firefly system [11], object composition petri nets (OCPN) [52], enhanced interval-based temporal relations [96], active and reactive object hierarchies [65], time-line specification using scripts [95], tree-based specification of serial and parallel requirements using

synchronization markers [85], hierarchical object composition in the CWI multimedia interchange format [13], dynamic timed petri nets and adaptive OCPN [72], conditional and temporal dependencies [18], and by mapping path expressions to timed petri nets [39], and the virtual time line models of MHEG and HyTime.

Since the object and specification layers deal with the high-level programming abstractions related to authoring synchronized multimedia presentations, and since the proposed work is primarily concerned with system-level architectural issues, we will focus our efforts on the media and stream layers. Therefore, we will not discuss the specification layer any further, and we will only be concerned with the user interaction portions of the object layer. However, we will provide appropriate interfaces so that more fully developed object and specification layers can benefit from our work.

4.4.5 Human Factors

Despite the best efforts of a system designed to support and enforce media synchronization, violations will occur. Thus, it is important to be aware of the levels of human tolerance to such errors. Reaction time to head tracking must be low in order to avoid user illness, and the delay between field-sequential display of stereo pairs must be short enough to enable comfortable fusion. In addition synchronization between related streams of audio, video, text, and graphics must be maintained so that users do not become annoyed. Human perception studies are reported in [68] and [88], and we summarize the relevant values in Table 1. A synchronization support system can use these QoS parameters to define a range of acceptable behavior and dynamically adapt to environmental changes.

Media	Synchronization Class	Tolerance Level
Audio and Video	Lip synchronization	±80ms
Audio and Audio	Tightly coupled (stereo)	±11μs
Audio and Audio	Loosely coupled (multi-party conversation)	±120ms
Audio and Text	Annotation	±240ms
Video and Image	Overlay	±240ms
Video and Image	Non-overlay	±500ms
Video and Text	Overlay	±240ms
Video and Text	Non-overlay	±500ms
Video	Head tracking reaction time	60ms
Stereo Video	Field-sequential stereo delay	50ms

Table 1 Human tolerance levels to media synchronization errors.

4.4.6 Support Technologies

One of the unique aspects of the synchronization problem is that it crosses several domain boundaries. Issues from the device, system, and application areas are all relevant. In this section we review some of the research performed in other areas that can be useful in enforcing synchronization constraints. It is important to realize that the technologies described below, alone or in combination, do not solve the media synchronization problem. They merely provide facilities that would be useful for systems designed to support synchronization.

Clock Synchronization

The fundamental reason the synchronization problem exists is the fact that any multimedia presentation environment encompasses multiple clock domains. For example, the simple case of displaying a single stream of video with correct intra-stream synchronization can involve three separate physical clocks: the CPU clock, the frame buffer clock, and the time-of-day clock. Each clock has its own physical properties and flaws that cause it to deviate from the others. A good solution to the media synchronization problem must correctly deal with all relevant clocks in the system.

In a networked environment this problem is particularly pronounced, and a great deal of research has been conducted to investigate methods for ensuring distributed clock synchronization. The network time protocol (NTP [57]) is perhaps the best known and most widely used technique. It is capable of synchronizing time-of-day clocks on machines interconnected by wide area networks to within a few milliseconds with a stability of a few milliseconds per day [58]. With some simple hardware and software enhancements, this protocol can be improved to provide sub-millisecond accuracy for machines directly connected to a precision time reference [59]. Modern quartz oscillators are quite stable, exhibiting frequency changes that vary roughly 1–2ppm/ C, which amounts to less than a millisecond of error over the course of a day [92]. Thus, traditional clocks and a protocol such as NTP can provide an accurate time base for distributed multimedia presentations.

It is important to realize, however, that clock synchronization alone does not guarantee media synchronization. As mentioned above, there are several clocks involved in a multimedia presentation environment, but clock synchronization protocols only deal with time-of-day clocks. Further mechanisms are required to ensure that I/O device clocks are synchronized, but it is insufficient to synchronize all clocks at the display site alone. For media such as audio and video, the display period for MDUs must match the period with which the MDUs were recorded. Synchronizing the display and record site clocks will yield a correct presentation, but this is unnecessary and impractical [73]. A better solution is to maintain separate clock domains at the record and playback sites.

It is tempting to believe that simple extensions to standard clock synchronization algorithms (so that all relevant clocks are included) will provide excellent media synchronization, but this is a naive viewpoint. Poor synchronization quality is due to a combination of many factors ranging from resource contention to non-deterministic software overhead. Thus, clock synchronization is a powerful tool, but it is not a complete solution.

Operating System Support

Since the operating system (OS) serves as the interface between applications and computer hardware, it is naturally an area of active research for providing multimedia support. Synchronized media presentation systems can benefit from much of this work. A good introduction to some of the general issues can be found in [63]. Adequate facilities exist for time-independent media objects, and so the major challenges is to support time-dependent objects. This requires a means for applications to specify timeliness constraints to the OS, as well as time-driven management of all resources. Current systems typically rely on excess resources, which is clearly impractical. Instead, high resource utilization should be maintained, and performance should be degraded gracefully in the presence of overload conditions. Several different aspects of operating systems are being investigated in attempts to provide this type of multimedia support. We briefly mention some of this work below.

Processor scheduling is a major area of investigation. In [62] it was shown that current time-sharing schedulers augmented with static priority real-time support are incapable of yielding the desired behavior for multimedia applications. Only through trial and error could effective choices for priority settings and scheduling classes be found so that a particular application mix performed adequately. This work quantitatively demonstrated that additional support is required, and several scheduling algorithms have been proposed, e.g. earliest deadline first (EDF), rate monotonic, least laxity first, shortest job first, and deadline monotone [89]. Variations and enhancements to these algorithms have also been proposed, but no approach has thus far distinguished itself as being the most appropriate.

Proper resource management is another OS issue which must be properly addressed. This includes the translation of application-level specifications of QoS constraints into specific resource requirements, negotiation between disparate resources to agree upon common QoS parameters, admission control, as well as reserving appropriate resources to ensure that QoS guarantees can be met [60]. For media streams rate control and buffer management must also be supported. Dynamically adapting to changing conditions requires that resource utilization be monitored and usage patterns altered at run time. Such efforts can provide excellent support to the stream layer in the synchronization reference model outlined above.

File system support for multimedia has also received a great deal of attention recently [35]. One major issue is the development of appropriate disk scheduling algorithms, such as EDF, Scan-EDF, round robin, and the grouped sweeping scheme. These algorithms provide varying degrees of complexity and performance quality. Data layout is another area of concern, and common schemes include contiguous placement, constrained placement, log-structured placement, data striping, and data interleaving. Since multimedia data must often be retrieved from secondary storage in a time-critical fashion, such research is significant.

Network Support

The networking issue which most directly relates to media synchronization is connection QoS guarantees, i.e. delay and jitter control. This requires the ability for users to specify desired parameters such as throughput, minimum and maximum delay, tolerable jitter, and allowable error rate [49]. A connection is then established which attempts to ensure that the constraints are met [6], [29], [48]. Resource reservation can provide deterministic service, while statistical guarantees provide the user with a certain probability that the parameters can be maintained. A protocol scheme implemented within the network can provide bounded delay and jitter end-to-end [28]. Finally, at the receiving site buffer management is required. Several approaches are possible, each with trade-offs in terms of jitter tolerance and introduced latency [91]. The combination of these techniques into a complete network service can thus provide performance guarantees for time-critical media streams. This is extremely useful for implementations of the reference model stream layer, since it significantly eases the problems associated with intra-stream synchronization in a distributed environment.

4.4.7 Current Techniques

Many techniques have been proposed to address various aspects of the media synchronization problem. In this section we investigate some of these methods, focussing primarily on media and stream layer solutions (see the reference model described in Section 4.4.4).

One common technique for achieving inter-stream synchronization is to interleave the associated media streams. QuickTime, CD-I, DVI, MPEG I, and MPEG II all utilize this approach to manage lip synchronization for audio and video[3]. However, there are several drawbacks to interleaving ([43], [49], [61]). It forces all streams to originate from the same storage subsystem or network connection, thereby significantly decreasing system flexibility and precluding the possibility of synchronizing media from different providers. All streams must also have the same QoS and priority level. Hence, efficient use of stream-specific information cannot be made. Congestion control mechanisms which drop data from a single medium, such as video, are also not possible. Errors in the interleaved data affect all the media streams, and scalability is poor. Multiplexing and de-multiplexing the different streams can be quite costly, particularly when different variable bit rate coders are used. The primary appeal of this approach is the belief that synchronization is straightforward for such multiplexed data. However, a system designed to support MPEG I system level synchronization with up to sixteen video and thirty-two audio streams required substantial effort to enforce synchronization once the streams were de-multiplexed, and the resulting quality was still poor [10]. Thus, although it is frequently used in stand-alone environments, such as personal computers, interleaving is widely regarded as a poor choice for synchronization support.

The Continuous Media Player [77] was developed at the University of California, Berkeley. The player is designed to display audio and video streams with correct lip synchronization. The streams are located on a single server machine, and presentation is performed at a remote client. Using NTP, the time-of-day clocks on the two machines are synchronized so that data transmission and retrieval are coordinated. The client sends feedback to the server so that the data rate can be adapted to meet the currently available resources. A one second buffer at the client is used to smooth network delay and jitter. The Continuous Media Player is somewhat ad hoc, and the result is a highly application-specific solution to the synchronization problem. In addition a one second buffer is a highly expensive use of memory capacity. Synchronizing the clocks of the client and server will not always be possible [73], and the clock rates of the frame buffer and audio codec are not taken into account. Hence, although this approach is not generally applicable, the use of adaptive feedback to dynamically control the data rate is quite useful.

Synchronization servers can be used to enforce intra-stream and inter-stream synchronization in distributed environments. This approach is designed to support multiple display sites. If the synchronization server also acts as a media server [73], a single data source can be used. Multiple data sources are possible when the synchronization server is simply a stream manager located within the network [86]. The synchronization server is responsible for characterizing network delays in order to determine relative clock offsets at the display sites. Periodically, the display sites transmit feedback information, such as the currently displayed and most recently received frames, to the server. This data is utilized to compute the likely intervals during which individual MDUs are displayed. When synchronization has been lost, the server instructs the display sites to pause or skip certain streams to regain proper playback. A major problem with this approach is the bottleneck the synchronization server represents. Since all streams must pass through the server, a system which utilizes this approach will not scale to the large numbers of streams required by the proposed system. In addition choosing when to re-synchronize the streams is an error-prone process. Adjustments may be out of phase with the current situation, or they may even exacerbate the problem.

3. MPEG II also supports streams which are not interleaved.

The Acme I/O server [3] is designed to support intra- and inter-stream synchronization. Media objects may originate from multiple sources, but they must be displayed at a common site. Synchronization is provided through the abstraction of a logical time system (LTS). An LTS has a current value which increases at about the same rate as real-time with a resolution of a few tens of milliseconds. Each MDU is assigned a sequence number which represents a timestamp in the LTS. MDUs with the same time stamp must be displayed at roughly the same point in real time. An LTS may advance according to the clock rate of an I/O device (device-driven), or it may advance according to the rate of data flow on a network connection (connection-driven). Each I/O device is bound to an LTS. Hence, I/O on devices within the same LTS will be synchronized. However, the I/O rate of a particular device may not match the rate of the LTS to which it is bound. For example, to display synchronized audio and video, the audio clock may be used to derive the LTS, but video will not be displayed at that same rate. To address this rate matching problem, a scheme for pausing a stream and skipping MDUs is utilized. The Acme server thus provides a nice abstraction and mechanism for local synchronization. Support is not available for multiple display sites, and no feedback mechanism is utilized to control the data rate across a network.

Another scheme to synchronize the presentation of one or more media streams is presented in [53]. Media data may originate from multiple remote locations, and synchronization is enforced at a single display site. Before presentation begins, a communication session is established, and various parameters are defined, e.g. the starting time, bandwidth and buffer space requirements, delay tolerance, etc. Three processes control the display of the media streams. The transmission and retrieval process is responsible for transmitting data from the remote sites at appropriate times, as well as appending incoming data to a buffer queue. The monitor and control process monitors the buffer queues and initiates recovery actions when underflow or overflow is imminent, i.e. skip or pause a stream. It is also responsible for adjusting queue levels so that streams remain synchronized to each other or to a time reference. The playout process continuously retrieves the topmost MDU from each queue and displays it at the appropriate time. This represents a primitive buffer management scheme that does not support multiple display sites and does not provide end-to-end feedback. Predictive techniques to help prevent buffer overflow or underflow are also not used. More advanced schemes are described below.

Maintaining buffers of media data at the display site is an effective means of smoothing transient network delays, but end-to-end feedback is necessary to reduce the possibility that overflow or underflow will occur. The schemes described in [41] and [5] utilize such feedback for rate control across a network. Buffer levels are monitored, and when certain thresholds are crossed, a request for corrective action (speed up or slow down) is sent to the transmitter. In addition the system in [5] recognizes that media data often requires application-specific processing before being displayed, e.g. video decompression. To support this type of scenario a two step approach to synchronization is used. First, when the application requests an MDU, a coarse grain synchronization step is performed. This may involve pausing or skipping the stream to ensure the proper MDU is passed to the application. Then, when processing is complete and the application is ready for display, a fine grain synchronization step is performed. This involves suspending the process until the proper display time has been reached. Although these schemes do not represent complete stream-level solutions, they provide useful insight into the need for proper buffer management and adaptive flow control.

The Flow Synchronization Protocol [27] and the Adaptive Synchronization Protocol [76] represent sophisticated stream management schemes that support completely distributed sources and sinks for media data. The time-of-day clocks on all machines involved in a multimedia presentation are synchronized with a protocol such as NTP. This is useful for estimating the total delay incurred

at all sites for the collection, transmission, and display of media data. Control messages are exchanged so that a total end-to-end delay for all streams can be agreed upon, and subsequent MDUs are then delayed up to this amount before being displayed. In this way latency is equal across all streams, and intra- and inter-stream synchronization are thus provided. Although these protocols are quite robust and effective, they do not handle overload conditions gracefully, instead relying on better OS support in the future. In addition they estimate the data capture and presentation delays without specifically taking the clocks associated with the I/O devices into account. This can cause MDUs to be displayed at incorrect times so that synchronization is no longer achieved.

The Orchestration Service [15] is a stream-oriented synchronization mechanism for distributed environments. A particular stream is chosen as a master, and global clock synchronization is utilized to ensure that the other slave streams progress at the same rate as the master. When a stream gets behind, MDUs are skipped so that it can catch up, and the stream is blocked when it runs ahead of schedule. The system relies on underlying support from the network to guarantee media data will be delivered at a given rate and with a specified delay. Media data are buffered to ensure smooth playback. However, this approach does not address the clock rates of individual devices, and no end-to-end feedback is provided to dynamically control the presentation rate in response to changing conditions.

The Multimedia Objects in a Distributed Environment (MODE) system [7] supports synchronization based on reference points. Media data may originate from multiple sources but is displayed at a single site. MODE includes a synchronization specification tool, as well as a local synchronizer to control presentations. For each media stream to be displayed, the synchronizer creates a thread to manage its intra-stream synchronization. These threads may execute at different priority levels. A synchronization point corresponds to reference points in two or more streams which must be simultaneously displayed. A signalling mechanism is used to enforce synchronization. When a presentation thread reaches a synchronization point, it signals to all other threads involved in the synchronization point. If a thread receives such a signal and is behind, it may accelerate its presentation or skip ahead to catch up. Meanwhile, threads which have already arrived at the synchronization point may pause their corresponding streams while waiting for other threads. This is a somewhat simplistic method which relies heavily on adequate operating system support. In addition end-to-end feedback is not provided, and there is no means for degrading service gracefully when sufficient resources are not available.

Tactus [19] is another system designed to synchronize media data in a distributed environment. All media objects are displayed at a single site, but the may originate from multiple remote locations. Data are computed in advance of real-time, and a timestamps indicating the desired presentation times are attached. Global clock synchronization is utilized so that timestamps are consistent across all involved participants. The data are transmitted to the display site, where they are buffered by the Tactus server. Finally, data are dispatched to I/O devices according to timestamps. No provisions are made for buffer overflow, but when underflow occurs, the server stalls all media output until the situation is resolved. Feedback is then sent to the data sources to indicate the duration of the stall, and they may take action as appropriate. Tactus assumes sufficient network and computing resources are available, and so it does not handle error states gracefully.

Finally, it is important to realize that choosing an appropriate synchronization mechanism requires a means of evaluating its performance. A system capable of directly observing video displayed on a monitor along with sound from a speaker and quantitatively measuring synchronization errors is described in [80]. This system yields highly accurate and objective results

and is thus useful for comparing the performance of different synchronization schemes. This work also demonstrated that taking the clock rates of display devices into account is extremely important, sometimes more critical than proper scheduling support from the OS. Such external measurements are also useful for determining the time between delivery of MDUs to a display device and their actual presentation, which enables a system to make more accurate predictions of display latency. In addition this system is useful for isolating and quantifying the causes of synchronization errors, thereby guiding the development of adequate solutions.

5 Technical Approach

Rigorously designing and implementing the architecture depicted in Figure 3 requires more than simple engineering effort. Trade-offs between different components are inevitable, but such compromises can only be made through careful study and understanding of various system interrelationships. In this section we outline the approach we will adopt to gain the necessary insight to create an effective specification and realization of the system outlined in Section 2.

5.1 Camera and Display Configuration

Stereopsis, motion parallax, and occlusion are the most salient depth cues [98], and a high quality system should be able to support all three. A single camera provides occlusion, and a second camera adds stereopsis. Motion parallax, however, can only be supported with multiple cameras, and that provides the motivation to use an array of cameras in the proposed system. Since the amount of allowed motion increases with the number of cameras, as many as possible should be utilized. We plan to use on the order of sixty-four cameras to implement the proposed architecture. This represents a balanced trade-off between the need for many cameras and practical constraints, such as limited bandwidth and computing resources.

Many camera arrangements are possible. They could be placed horizontally in the parallel axes configuration, radially in a verged setup, or perhaps in a combination of these two in which pairs of parallel axis cameras are verged. Multiple rows of cameras could also be supported. In these configurations the aggregate field of view is effectively in one direction so that an observer experiences the effect of peering through a virtual window. Alternately, a more immersive experience can be obtained by arranging the cameras in a divergent configuration so that the optical axes are non-parallel and non-intersecting (see Figure 4). A circular setup would support a full 360 lateral field of view, and a semi-spherical arrangement would provide the experience of looking around within a virtual dome.

Similarly, various types of displays are available to view the video data from the cameras. A standard computer monitor can only display monoscopic video, but with the support of active stereo shutter glasses, time-multiplexed field-sequential stereoscopic sequences can also be presented. A polarizing projection system and passive filter glasses is appropriate for situations in which many viewers must occupy a single location, such as a theater setting. A lenticular display is autostereoscopic and can support multiple horizontally interlaced perspectives as well as multiple viewers. An HMD provides a much wider field of view and is thus more immersive, but this comes at the cost of poor resolution.

It is interesting to note that not all combinations of camera configurations and display interfaces are advisable. For example, when an HMD is used, viewpoint changes are directed by head rotations. Since the parallel axes configuration primarily supports lateral translation, it would make a poor choice. However, a divergent camera configuration supports such rotations and is therefore well

suited to this situation. Broadly speaking, viewpoint changes for immersive display types are effected by head rotations, whereas head translations are used for non-immersive displays. We choose to support non-immersive displays such as the Virtual Holographic Workstation [20], which provides stereoscopic viewing on a standard monitor as well as head tracking capabilities. Such a system offers excellent resolution and integrates easily with the normal computing environment. This limits us to "virtual window" camera arrangements such as the parallel axes configurations, but no *a priori* assumptions will be made regarding the nature of the camera configuration or display type within the system. Thus, any other combination would be equally well supported.

Given the above design choices the next step is to determine the best camera configuration to utilize. To accomplish this task we will conduct a user study to measure head motion. Although some anecdotal evidence suggests a range of one to two inches both horizontally and vertically [64], we propose a more detailed analysis. Users will be asked to observe a scene through an actual window while they are wearing head tracking apparatuses. Head position data will be recorded over time and used to determine the camera configuration needed to support the full range of motion, as well as the density required to minimize the flipping effect described in Section 4.2.1. It may turn out that the ideal configuration is not practical, and such experimental data will be extremely helpful in evaluating the costs of making compromises. Later sections will explore some of the other useful information that can be derived from this experiment.

Knowledge of the range of typical head motion is not sufficient information to determine an appropriate camera configuration. Other considerations must be taken into account. For example, verged cameras induce a certain level of vertical disparity into the perceived images. This makes it more difficult for binocular fusion to occur and consequently results in viewer discomfort [4]. In addition computing intermediate views between verged cameras requires an expensive process known as *rectification* to align the images in the same plane so that disparity can be computed [45]. The parallel axes configuration avoids these problems, but the simple geometry is potentially limiting. *Inter-pupillary distance* (IPD) is the distance between an observer's eyes and can range from roughly 53mm to 73mm with an average of 65mm. Truly accurate display of stereoscopic images requires that the IPD for an individual viewer be taken into account. However, studies have shown that IPD need not be exact, just not extreme. Human performance on depth related tasks was measured to be virtually the same for IPD values ranging from about 30mm to 80mm [75]. Hence, some flexibility is possible with regard to viewpoint separation. A view is *orthoscopic* if the objects appear "life size." Although non-orthoscopic views allow more flexibility in camera placement, they introduce perspective distortions that and are considered unacceptable by some. Ikehara and Cole [42], however, demonstrated that human performance on remote tasks did not diminish when non-orthoscopic views were utilized. Finally, head tracking will be an important component of the display configuration, and its use will have impact on the camera configuration to be utilized. For example, for non-head-mounted displays, head orientation is not useful. Perspective changes induced by head rotations are not intuitive, and so only translations should be considered [2]. In addition, the effects of head motion should be exaggerated, i.e. users prefer large viewpoint changes in response to small head movements.

All these considerations will be taken into account in order to determine the appropriate camera and display configuration for the proposed architecture.

5.2 Coding Scheme

The proposed system is an ambitious project if for no other reason than the enormous bandwidth demands it makes. The data rate for sixty-four full-size 30Hz raw video streams can be as high as 13.2Gbps, by far outstripping the capabilities of current technology. Clearly, some form of compression is required. A naive approach would be to utilize commercially available technology (i.e. JPEG, MPEG I, or MPEG II codecs), and compress each stream individually. Assuming an average bit rate of 6–8Mbps per stream, bandwidth requirements would be reduced to 384–512Mbps, which is well within the realm of possibility. There are several reasons that such an approach is suboptimal, but perhaps the most compelling is that each of the schemes mentioned above is far too costly to actually implement. A typical JPEG compression board capable of both encoding and decoding costs around $5000. An MPEG I encoder sports a $50,000 price tag, while decoders can be purchased for $300–$500. MPEG II coders are available for about $80,000, and decoders are roughly $1000 a piece. Clearly, using sixty-four separate encoders and two or more decoders for each display site is not cost effective. Another coding alternative must be found.

To help in the design and evaluation of a suitable coding scheme, it is important to understand the constraints that it must satisfy. We outline these requirements below:

- High Quality — Minimizing artifacts introduced by compression is a goal common to all coding schemes. However, with stereo video it is of particular concern, since decreased image quality can lead to discomfort and even illness over prolonged viewing periods. As outlined in Section 4.2.1, there are several human factors which must be taken into account when devising an appropriate scheme, but perhaps the most important is the need to evaluate the final quality subjectively. Quantitative measures are insufficient to determine the ease with which fusion can be achieved, how well depth related tasks can be performed, or the level of viewer discomfort. Thus, it will be necessary to perform user studies to assess the efficacy of the chosen technique.
- Moderate Bandwidth — Most coding algorithms attempt to remove as much redundancy as possible from the original video data. This can lead to extremely efficient techniques which achieve substantial savings in bandwidth. However, such dramatic reductions in the required data rate usually comes at the expense of other desirable features, such as quality or complexity. Therefore, we relax the bandwidth constraint to simply require that it be low enough to be adequately supported by current technology.
- Low Computational Cost — Since a great number of video streams are being utilized, it is important that the computational cost be kept to a minimum. As algorithmic complexity rises, implementing a given approach on current hardware so that it executes in real time becomes an increasingly greater challenge. Substantial engineering effort and computing resources can be required. This is simply not practical for such an ambitious system. In addition low computational requirements increases the range of possible destinations to include lower performance computers.
- Low Implementation Cost — As mentioned above, current implementations of standard compression algorithms are extremely expensive. This cost can be amortized by dividing it among a large number of users as in direct broadcast satellite television systems. Although the proposed architecture is only limited in scope by the size of the network, we envision a moderately sized user community. Thus, to make the system practical, it must be cost effective.

- Error Resilience — Fiber optic cables are emerging as the next generation in network transmission carriers. Bit errors are extremely rare for this high bandwidth medium. However, most protocols intended to run on such networks employ some form of packet switching. During times of congestion switches may become overloaded and discard excess packets. Thus, in order for a coding scheme to be well suited to such an environment, it must be resilient to packet loss. Algorithms such as JPEG and MPEG which utilize variable length coding are particularly susceptible to this type of error. Unless additional measures are taken, a single packet loss can result in the loss of several consecutive frames. Forward error correction can help address this problem, but it adds redundancy back into a video stream after a compression scheme worked hard to remove it. This is counterproductive, and a better approach is to support partial image reconstruction and fill in the missing data using some heuristic.
- Low Memory Requirements — This condition is related to the goal of low implementation cost. With so many streams being coded, the memory requirements can be extremely high. If that memory takes the form of fast SRAM, the expense can be large. The drawback of utilizing only small amounts of memory is that the number of frame stores is limited, thus constraining the algorithmic possibilities. For example, if intra-stream temporal predictions were desired, a frame store of roughly 900KB per stream would be required. This may represent too high a cost to warrant its use.
- Low Latency — As mentioned previously, we envision head tracking to be the primary mechanism by which observers will make viewpoint changes. To avoid motion sickness problems, these changes must occur in a timely fashion. Thus, the latency associated with switching to decompress a new stream must be minimized. This may preclude the use of temporal prediction since switching streams could require a reference frame to be decompressed before the desired frame which is derived from it. This increased work could result in the tolerance threshold being exceeded.
- Scalability — Since the proposed architecture is aimed at supporting future applications which require many streams of video, it is important to realize that the number of streams applications will need can grow. Thus, the coding scheme must be easily scaled to incorporate larger and larger numbers of streams.

Most compression schemes are geared towards minimizing bandwidth requirements while maintaining acceptable quality. It should be clear, however, that complexity and cost must be given larger consideration in the context of the proposed architecture. Thus, we will incorporate these concerns into a cost-benefit analysis methodology to measure the performance of the chosen coding technique. This will enable us to evaluate the effects of different design trade-offs so that reasonable compromises can be made. In addition subjective user studies will be conducted as a complement to this measurement scheme.

As shown in Figure 1, compression is just one aspect of the video-network interface. First, the NTSC signals must be decoded and digitized. Then, the digital data must be compressed, and the streams may be aggregated into groups either for easier transmission or due to inter-stream dependencies. Finally, the streams will be packetized and delivered to the network for transmission. We will investigate various design choices to determine the best architecture for the video-network interface. For example, if a single monolithic entity will be responsible for all these functions, the construction of custom hardware will be required. Alternatively, several workstations with stock video capture boards could be used in concert to provide a more decentralized approach. This has the

benefit of allowing the system to be scaled incrementally. For this method to be feasible, however, the coding scheme must be sufficiently straightforward that software implementations will execute in real time. In addition the number of streams coded per machine must be relatively high to make the system cost effective. The final architecture must represent an artful combination of various system components so that a careful balance between the various requirements listed above is achieved.

5.3 Network Design

In order to adequately support the proposed system, the network must provide certain functionality. We outline some of the essential networking requirements below:

- QoS Guarantees — Most current networks provide a best-effort service, but media data possess time constraints that must be satisfied to ensure proper presentation. Hence, the network must possess the ability to transmit time-critical traffic with a reasonable probability of success.
- High Bandwidth — Since we assume numerous media streams will be used by individual applications, the bandwidth demands on the network will be substantial. In addition the bandwidth requirements for a server and its clients may be non-uniform, since each client may only receive a small subset of the streams transmitted by a server.
- Low Latency — Since media data possess timeliness requirements, providing fine-grain interaction requires the ability to make modifications extremely quickly. In the proposed architecture, some of these interactions must take place within the network, thus requiring low latency.
- Multicast — The nature of the proposed system implies the need for multicast. Since a server will transmit its set of media streams to a potentially large group of clients, efficient utilization of network resources demands integral support for multicast.
- Fast Signalling — In the proposed architecture fine-grain connection management is needed in the form of dynamic modifications to the set of streams being received by a client. Since these alterations must occur within the time constraints dictated by the media data, a fast signalling mechanism is required to make the necessary changes to network state.
- Integrated Traffic Management — The proposed system requires advanced support for the transmission of numerous media streams, but other applications with dramatically different requirements may be using the same network. Thus, integrated support for multiple types of traffic must be provided by the network.

We will investigate various types of networks (e.g. ATM, FDDI, IP-based, etc.) and analyze their abilities to meet the above goals, and we will evaluate the dynamic multicast performance of different switches/routers. Based on our analysis, we will choose the most appropriate network technology to use as the basis for the proposed system, as well as the best method for implementation. This will involve a detailed study of the design criteria and performance of modern networking hardware. Then, we will implement a complete system by augmenting current technology to provide the added functionality required by the proposed architecture, including the fast signalling mechanism and multicast feedback coordination.

Transmitting video streams requires close interaction between the packetization scheme and the compression algorithm. For example, a hierarchical coding scheme may require the use of multiple layers, each sent at a different priority, and the error concealment/correction scheme also has a

significant impact. Packetization can be a large source of delay, so this must be done carefully. The coding and packetization will be developed in close association to ensure that both compression and network transmission can be performed efficiently.

We will conduct an empirical study to determine the best place within the network to perform the dynamic modifications necessary for the fine-grain connection management. This involves a trade-off between latency and efficiency, i.e. it is more efficient to switch between streams as close to the source as possible, but the cost is increased latency as the changes are made farther from the destination. Our experiments will help to reveal the relationship between latency and efficiency so that the optimum trade-off can be made.

5.4 Synchronization Support

As mentioned above, synchronization is widely regarded as a fundamental problem in multimedia research. Within the scope of this project, we do not attempt to provide a complete solution. We are primarily concerned with system-level (media and stream layers, see Section 4.4.4) issues related to supporting distributed, multi-party, interactive applications. In this section we discuss our technical approach for providing such support.

5.4.1 Media Layer

At its heart media synchronization is a resource management problem. I/O devices must be scheduled appropriately so that MDUs are collected or displayed at the proper times. Hence, to provide the necessary support for synchronization time-driven resource management (TDRM [63]) must be used for all devices. Utilizing TDRM requires extending the media layer of the synchronization reference model to make the temporal component explicit.

This involves changing the primitives to read time-dependent media data from I/O devices to return the time at which the MDUs were captured. In addition the commands to display time-dependent media data should include an indication of the ideal presentation times of the MDUs, and appropriate implementations must ensure that these time constraints are satisfied. In this way a fully distributed solution is supported since timing control is placed as close to the devices as possible. Centralized servers, which are commonly used, are inherently flawed since they relinquish control too soon.

At this level of abstraction all timing information is known, i.e. the capture and presentation times of MDUs can be uniquely determined. Thus, a global time line is the simplest and most effective time model to use as the basis for the read and write primitives. A single time line may be shared by multiple media objects so that inter-object synchronization can be easily achieved. However, creating such a synchronization domain requires the devices involved to synchronize their clocks, which can be difficult for domains which cross machine boundaries. This problem can be addressed hierarchically by synchronizing the clock of each I/O device to the time-of-day clock on its host, and a network clock synchronization protocol, such as NTP, can be used to align the time-of-day clocks on different machines. These mechanisms can be used to accurately determine the acquisition times of individual MDUs, as well as to guarantee timely display of the MDUs from a multimedia presentation.

5.4.2 Stream Layer

At the stream layer timing information is also well known, and so a global time line is an adequate time model. Based on user input the client-side stream layer will select which streams are to be displayed, as well as the starting and stopping points for these presentations. As a result, MDUs will be delivered to the media layer with a timestamp indicating the value of the time line at which it should be displayed.

To smooth transient timing anomalies streams will be buffered prior to presentation. Possible buffering schemes include the ones described in [27] and [76], but a good solution must also include end-to-end feedback to provide the necessary rate control. A software phase locked loop strategy, such as the one in [16], is an excellent method of feedback. We will adapt such a model to include the proper coordination and support for multicast connections in which multiple clients are transmitting feedback upstream to a single server. The server-side stream layer will respond to such feedback by performing the necessary rate control functions.

The client-side stream layer must also provide continued support in the face of insufficient resources. This involves local decisions to skip or pause a stream, as in [3], so that transient overload conditions can be tolerated, as well as high-level feedback for persistent trouble. Inter-object synchronization is supported by attaching appropriate presentation timestamps to MDUs from related streams.

Since the assumed model of interaction is for users to switch between viewed streams at a fine-grain, a neighborhood of streams will be buffered at the client. This allows the system to minimize the latency of switching streams by avoiding network transactions simply making a local change. The application is responsible for changing the set of buffered streams, which requires the stream layer to make the appropriate requests from the network to change multicast group configurations. The cost of this optimization is buffer space, and this allows us to trade latency for bandwidth. Clients which are close to the server will buffer fewer streams than more distant clients, which reflects the increasing cost of network operations as hosts become further separated.

5.4.3 Object and Specification Layers

In our system we do not explicitly provide object and specification layers. Instead, the client and server applications interact directly with the lower layers, which are designed to provide sufficient abstraction for other object and specification layers, such as the ones described in Section 4.4.4, to be built on top.

The client and server applications provide standard global coordination facilities, but the main synchronization-related activities occur at the client. These activities deal with switching between viewed streams in response to input from a head-tracking device. Using information from the head motion study described in Section 5.1, the typical head velocity can be determined. This information allows the application to choose an appropriate number of streams to buffer so that most stream switches can be simple local display changes which avoid network traffic. The application directs stream switches and requests changes in the composition of the buffered set so that a window of streams is maintained around the user's current head position. These buffer changes result in requests for modifications to multicast group configurations. To further minimize the latency of these perspective changes, an algorithm to predict future head location (such as the one in [32]) can be used, and the data from the head motion study will help develop and verify this algorithm.

6 Contributions

Designing and implementing an architecture for distributed, interactive, multi-stream, multi-participant audio and video represents a substantial undertaking, and several important contributions result. First, a novel system capable of supporting the demands of future multimedia applications will be constructed. This system provides a springboard for further investigation of other topics, such as computer vision, virtual reality, remote collaboration, etc. A key feature is the assumption of insufficient resources, and the system is designed to provide adequate support and graceful service degradation for that type of environment. Building such a complex system requires numerous trade-offs and compromises, which can only be made by taking into consideration the multi-disciplinary goals of the architecture.

At the application level several advances will be made regarding the use of multi-viewpoint audio and video. Various camera configurations will be investigated, and one will be chosen which best captures the range of user head motion. In addition an efficient and cost-effective coding scheme will be developed to compress the video streams. This compression technique will exploit the high degree of spatial redundancy between streams while preserving high fidelity, and it will be scalable to large numbers of streams.

Support for dynamic and fine-grain network multicast provides another major contribution of this work. Multiple streams will be transmitted to a group of receivers who accept different subsets of those streams. The model of interaction assumes that receivers are switching between streams frequently and often, thus requiring fine-grain connection management in a time-critical environment. A fast signalling protocol will be developed to perform the multicast group modifications necessary to support this mode of interaction. In addition feedback from the receivers is needed to effectively control the flow of media data, but it is difficult to coordinate and utilize such feedback in a multicast connection, particularly when the receivers possess heterogeneous capabilities. A protocol will be designed to provide support for multicast feedback.

A system-level solution to the media synchronization problem is also proposed. This scheme will support multiple streams of audio and video, as well as fine-grain interaction with the streams. This interaction is provided in the form of frequent switching between viewed streams. Synchronization across multiple hosts will be supported, and multiple participants may simultaneously view the media streams. Mechanisms for graceful degradation will be provided so that transient timing disruptions and data errors do not result in loss of synchronization. In general resource management in current systems does not adequately support time-critical activities, and our solution to the synchronization problem provides time-driven resource control to ensure that media devices correctly capture and display time-dependent media data.

This architecture represents the next generation of multimedia systems. As such, it constitutes significant advance in multimedia research and technology, and we hope that it will inspire exciting new work in the future.

7 References

1. K. Ahlers, D. Breen, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer, “An Augmented Vision System for Industrial Applications,” *Telemanipulator and Telepresence Technologies, Proceedings of the SPIE*, vol. 2351, 1994, pp. 345–359.

2. R. Akka, “Utilizing 6D Head Tracking Data for Stereoscopic Computer Graphics Perspective Transformations,” *Stereoscopic Displays and Applications IV, Proceedings of the SPIE*, vol. 1915, 1993, pp. 147–154.
3. D. Anderson and G. Homsy, “A Continuous Media I/O Server and Its Synchronization Mechanism,” *IEEE Computer*, 24(10), October 1991, pp. 51–57.
4. A. Ariyaeeinia, “Distortions in Stereoscopic Displays,” *Stereoscopic Displays and Applications III, Proceedings of the SPIE*, vol. 1669, 1992, pp. 2–9.
5. F. Bastian and P. Lenders, “Media Synchronization on Distributed Multimedia Systems,” *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994, pp. 526–531.
6. R. Bettati, D. Ferrari, A. Gupta, W. Heffner, W. Howe, M. Moran, Q. Nguyen, R. Yavatkar, “Connection Establishment for Multi-Party Real-Time Communication,” *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, April 1995.
7. G. Blakowski, J Hubel, U. Langrehr, and M. Muhlhauser, “Tool Support for the Synchronization and Presentation of Distributed Multimedia,” *Computer Communications*, 15(10), December 1992, pp. 611–618.
8. M. Bolas and S. Fisher, “Head-coupled Remote Stereoscopic Camera System for Telepresence Applications,” *Stereoscopic Displays and Applications, Proceedings of the SPIE*, vol. 1256, 1990, pp. 113–123.
9. J. Bolot, T. Turletti, and I. Wakeman, “Scalable Feedback Control for Multicast Video Distribution in the Internet,” *Proceedings of the SIGCOMM Symposium on Communications Architectures and Protocols*, August 1994, pp. 58–67.
10. J. Boucher, Z. Yaar, E. Rubin, J. Palmer, and T. Little, “Design and Performance of a Multi-Stream MPEG-I System Layer Encoder/Player Set,” *Multimedia Computing and Networking, Proceedings of the SPIE*, vol. 2417, March 1995, pp. 435–446.
11. M. Buchanan and P. Zellweger, “Automatically Generating Consistent Schedules for Multimedia Documents,” *Multimedia Systems*, 1(2), 1993, pp. 55–67.
12. J. Buford, *Multimedia Systems*, ACM Press, New York, 1994.
13. D. Bulterman, “Specification and Support of Adaptable Network Multimedia,” *Multimedia Systems*, 1(2), 1993, pp. 68–76.
14. J. Calvin, A. Dickens, B. Gaines, P. Metzger, D. Miller, and D. Owen, “The SIMNET Virtual World Architecture,” *Proceedings of the IEEE Annual Virtual Reality International Symposium*, 1993, pp. 450–455.
15. A. Campbell, G. Coulson, F. Garcia, and D. Hutchison, “A Continuous Media Transport and Orchestration Service,” *Computer Communication Review*, 22(4), October 1992, pp. 99–110.
16. S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole, “A Distributed Real-Time MPEG Video Audio Player,” *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, May 1995, pp. 151–162.
17. R. Cole and J. Merritt, “Remote Manipulation Tasks Impossible without Stereo TV,” *Stereoscopic Displays and Applications, Proceedings of the SPIE*, vol. 1256, 1990, pp. 255–265.

18. J. Courtiat, L. Carmo, and R. Oliveira, "A New Mechanism for Achieving Inter-stream Synchronization in Multimedia Communication Systems," *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994, pp. 173–182.
19. R. Dannenberg, T. Meuendorffer, J. Newcomer, D. Rubine, and D. Anderson, "Tactus: Toolkit-level Support for Synchronized Interactive Multimedia," *Multimedia Systems*, 1(2), 1993, pp. 77–86.
20. M. Deering, "Explorations of Display Interfaces for Virtual Reality," *Proceedings of the IEEE Annual Virtual Reality International Symposium*, 1993, pp. 141–147.
21. S. Deering, "Host Extensions for IP Multicasting," *Internet Request for Comments 1112*, Network Working Group, August 1989.
22. L. Delgrossi and L. Berger (Eds), "Internet Stream Protocol Version 2 Protocol Specification — Version ST2+," *Internet Request for Comments 1819*, Network Working Group, August 1995.
23. L. Delgrossi, C. Halstrick, D. Hehmann, R. Herrtwich, O. Krone, J. Sandvoss, and C. Vogt, "Media Scaling in a Multimedia Communication System," *Multimedia Systems*, 2(4), 1994, pp. 172–180.
24. I. Dinstein, M. Kim, A. Henik, and J. Tzelgov, "Compression of Stereo Images Using Subsampling and Transform Coding," *Optical Engineering*, 30(9), September 1991, pp. 1359–1364.
25. D. Drascic and J. Grodski, "Using Stereoscopic Video for Defense Teleoperation," *Stereoscopic Displays and Applications IV, Proceedings of the SPIE*, vol. 1915, 1993, pp. 58–69.
26. A. Dumbreck, E. Abel, and S. Murphy, "3D TV System for Remote Handling: Development and Evaluation," *Stereoscopic Displays and Applications, Proceedings of the SPIE*, vol. 1256, 1990, pp 226–236.
27. J. Escobar, C. Partridge, and D. Deutsch, "Flow Synchronization Protocol," *IEEE/ACM Transactions on Networking*, 2(2), April 1994, pp. 111–121.
28. D. Ferrari, "Design and Applications of a Delay Jitter Control Scheme for Packet-Switching Internetworks," *Computer Communications*, 15(6), July/August 1992, pp. 367–373.
29. D. Ferrari and D. Verma, "A Scheme for Real-Time Channel Establishment in Wide Area Networks," *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990, pp. 368–379.
30. S. Fisher, "Viewpoint Dependent Imaging: An Interactive Stereoscopic Display," *Processing and Display of Three-Dimensional Data, Proceedings of the SPIE*, vol. 367, 1982, pp. 41–45.
31. M. Frankel and J. Webb, "Design, Implementation, and Performance of a Scalable Multi-Camera Interactive Video Capture System," *Technical Report CMU-CS-95-162*, Carnegie Mellon University, June 1995.
32. M. Friedmann, T. Starner, and A. Pentland, "Synchronization in Virtual Realities," *Presence: Teleoperators and Virtual Environments*, 1(1), Winter 1992, pp. 139–144.
33. T. Fujii and H. Harashima, "Coding of an Autostereoscopic 3D Image Sequence," *Visual Communication and Image Processing '94, Proceedings of the SPIE*, vol. 2308, September 1994, pp. 930–941.

34. W. Gaver, G. Smets, and K. Overbeeke, "A Virtual Window on Media Space," *Proceedings of the Conference on Computer-Human Interaction*, May 1995, pp. 257–264.
35. D. Gemmell, H. Vin, D. Kandlur, P. Rangan, and L. Rowe, "Multimedia Storage Servers: A Tutorial," *IEEE Computer*, 28(5), May 1995, pp. 40–49.
36. P. Gunatilake, M. Siegel, and A. Jordan, "Compression of Stereo Video Streams," *Signal Processing of HDTV V*, 1994, pp. 173–185.
37. L. Harrison and D. McAllister, "Problems with Lossy Compression of Stereo Pairs," *Stereoscopic Displays and Applications III, Proceedings of the SPIE*, vol. 1669, 1992, pp. 39–50.
38. M. Hirose, K. Yokoyama, and S. Sato, "Transmission of Realistic Sensation: Development of a Virtual Dome," *Proceedings of the IEEE Annual Virtual Reality International Symposium*, 1993, pp. 125–131.
39. P. Hoepner, "Synchronizing the Presentation of Multimedia Objects," *Computer Communications*, 15(9), November 1992, pp. 557–564.
40. R. Horst, "A Digital Codec for 3DTV Transmission," *Signal Processing of HDTV IV*, 1993, pp. 489–495.
41. J. Hui, E. Karasan, J. Li, and J. Zhang, "Client-Server Synchronization and Buffering for Variable Rate Multimedia Retrievals," *Proceedings of the IEEE Workshop on Multimedia Synchronization*, May 1995.
42. C. Ikehara and R. Cole, "The Effects of Perspective Distortion in Stereoscopic Video Displays," *Stereoscopic Displays and Applications IV, Proceedings of the SPIE*, vol. 1915, 1993, pp. 49–57.
43. K. Jeffay, D. Stone, and F. Smith, "Kernel Support for Live Digital Audio and Video," *Computer Communications*, 15(6), July/August 1992, pp. 388–395.
44. R. Kalawsky, *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley Publishing Company, Wokingham, England, 1993.
45. S. Kang, J. Webb, C. Zitnick, and T. Kanade, "An Active Multibaseline Stereo System with Real-Time Image Acquisition," *Technical Report CMU-CS-94-167*, Carnegie Mellon University, Sept. 1994.
46. A. Katayama, K. Tanaka, T. Oshini, and H. Tamura, "A Viewpoint Dependent Stereoscopic Display Using Interpolation of Multi-Viewpoint Images," *Stereoscopic Displays and Virtual Reality Systems II, Proceedings of the SPIE*, vol. 2409, March 1995, pp. 11–20.
47. B. Kost and S. Pastoor, "Visibility Thresholds for Disparity Quantization Errors in Stereoscopic Displays," *Proceedings of the SID*, 32(2), 1991, pp. 165–170.
48. A. Krishnamurthy and T. Little, "Connection-Oriented Service Renegotiation for Scalable Video Delivery," *Proceedings of the International Conference on Multimedia Computing and Systems*, May 1994, pp. 502–507.
49. P. Leydekkers and B. Teunissen, "Synchronization of Multimedia Data Streams in Open Distributed Systems," *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1991, pp. 94–104.
50. L. Li, A. Karmouch, N. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 1 — Temporal Modeling of Collaborative Multimedia Scenarios," *Multimedia Systems*, 1(4), 1994, pp. 143–153.

51. L. Li, A. Karmouch, N. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 2 — Synchronization Algorithms," *Multimedia Systems*, 1(4), 1994, pp. 154–165.
52. T. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990, pp. 413–427.
53. T. Little and F. Kao, "An Intermedia Skew Control System for Multimedia Data Presentation," *Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1992, pp. 130–141.
54. J. Liu, "Stereo Image Compression — The Importance of Spatial Resolution in Half Occluded Regions," *Human Vision, Visual Processing, and Digital Display VI, Proceedings of the SPIE*, vol. 2411, April 1995, pp. 271–276.
55. J. Liu and R. Skerjanc, "Construction of Intermediate Pictures for a Multiview 3D System," *Stereoscopic Displays and Applications III, Proceedings of the SPIE*, vol. 1669, 1992, pp. 10–19.
56. M. Macedonia and D. Brutzman, "MBone Provides Audio and Video across the Internet," *IEEE Computer*, 27(4), April 1994, pp. 30–36.
57. D. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Transactions on Communications*, 39(10), October 1991, pp. 1482–1493.
58. D. Mills, "On the Accuracy and Stability of Clocks Synchronized by the Network Time Protocol in the Internet System," *Computer Communication Review*, 20(1), January 1990, pp. 65–75.
59. D. Mills, "Precision Synchronization of Computer Network Clocks," *Computer Communications Review*, 24(2), April 1994, pp. 28–43.
60. K. Nahrstedt and R. Steinmetz, "Resource Management in Networked Multimedia Systems," *IEEE Computer*, 28(5), May 1995, pp. 52–63.
61. C. Nicolaou, "An Architecture for Real-Time Multimedia Communication Systems," *IEEE Journal on Selected Areas in Communication*, 8(3), April 1990, pp. 391–400.
62. J. Nieh, J. Hanko, J. Northcutt, and G. Wall, "SVR4 UNIX Scheduler Unacceptable for Multimedia Applications," *Proceedings of the Fourth International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1993, pp. 41–53.
63. J. Northcutt and E. Kuerner, "System Support for Time-Critical Applications," *Computer Communications*, 16(10), Oct. 1993, pp. 619–636.
64. W. Paley, "Head Tracking Stereo Display: Experiments and Applications," *Stereoscopic Displays and Applications III, Proceedings of the SPIE*, vol. 1669, 1992, pp. 84–89.
65. M. Papathomas, G. Blair, G. Coulson, and P. Robin, "Addressing the Real-Time Synchronization Requirements of Multimedia in an Object-Oriented Framework," *Multimedia Computing and Networking, Proceedings of the SPIE*, vol. 2417, February 1995, pp. 190–201.
66. A. Park and R. Kazman, "Augmented Reality for Mining Teleoperation," *Telemanipulator and Telepresence Technologies, Proceedings of the SPIE*, vol. 2351, 1994, pp. 119–129.
67. C. Parris, H. Zhang, and D. Ferrari, "Dynamic Connection Management of Guaranteed-Performance Multimedia Connections," *Multimedia Systems*, 1(6), 1994, pp. 267–283.

68. S. Pastoor, "3D Television: A Survey of Recent Research Results on Subjective Requirements," *Signal Processing: Image Communication*, 4(1), November 1991, pp.21–32.
69. S. Pastoor and I. Beldie, "Subjective Assessments of Dynamic Visual Noise Interference in 3D TV Pictures," *Proceedings of the SID*, 30(3), 1989, pp. 211–215.
70. S. Pastoor and K. Schenke, "Subjective Assessments of the Resolution of Viewing Directions in a Multi-Viewpoint 3D TV System," *Proceedings of the SID*, 30(3), 1989, pp. 217–223.
71. M. Perkins, "Data Compression of Stereopairs," *IEEE Transactions on Communications*, 40(4), April 1992, pp. 684–696.
72. B. Prabhakaran and S. Raghavan, "Synchronization Models for Multimedia Presentation with User Participation," Multimedia Systems, 2(2), 1994, pp. 53–62.
73. P. Rangan, S. Ramanathan, and S. Sampathkumar, "Feedback Techniques for Continuity and Synchronization in Multimedia Information Retrieval," *ACM Transactions on Information Systems*, 13(2), April 1995, pp. 145–176.
74. K. Ritchey, "Image Based Panoramic Virtual Reality System," *Visual Data Interpretation, Proceedings of the SPIE*, vol. 1668, 1992, pp. 2–14.
75. L. Rosenberg, "The Effect of Inter-ocular Distance upon Operator Performance Using Stereoscopic Displays to Perform Virtual Depth Tasks," *Proceedings of the IEEE Annual Virtual Reality International Symposium*, 1993, pp. 27–32.
76. K. Rothermel and T. Helbig, "An Adaptive Stream Synchronization Protocol," *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1995, pp. 189–201.
77. L. Rowe and B. Smith, "A Continuous Media Player," *Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1992, pp. 376–386.
78. R. Sand, "3DTV Research and Development in Europe," *Stereoscopic Displays and Applications II, Proceedings of the SPIE*, vol. 1457, 1991, pp. 76–84.
79. R. Satava, "Telemanipulation, Telepresence and Virtual Reality for Surgery in the Year 2000," *Telemanipulator and Telepresence Technologies, Proceedings of the SPIE*, vol. 2351, 1994, pp. 162–171.
80. B. Schmidt, J. Northcutt, and M. Lam, "A Method and Apparatus for Measuring Media Synchronization," *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1995.
81. H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control," *Internet Draft*, Audio-Video Transport Working Group, file draft-ietf-avt-profile-06.ps, November 1995.
82. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," *Internet Draft*, Audio-Video Working Group, file draft-ietf-avt-rtp-07.ps, March 1995.
83. S. Sethuraman, M. Siegel, and A. Jordan, "A Multiresolutional Region Based Segmentation Scheme for Stereoscopic Image Compression," *Digital Video Compression: Algorithms and Technologies, Proceedings of the SPIE*, vol. 2419, April 1995, pp. 265–274.
84. C. Shaw and M. Green, "The MR Toolkit Peer Package and Experiment," *Proceedings of the IEEE Annual Virtual Reality International Symposium*, 1993, pp. 463–469.

85. D. Shepherd and M. Salmony, "Extending OSI to Support Synchronization Required by Multimedia Applications," *Computer Communications*, 13(7), September 1990, pp. 399–406.
86. S. Son and N. Agarwal, "Synchronization of Temporal Constructs in Distributed Multimedia Systems with Controlled Accuracy," *International Conference on Multimedia Computing and Systems*, May 1994, pp. 550–555.
87. R. Steinmetz, "Synchronization Properties in Multimedia Systems," *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990, pp. 401–412.
88. R. Steinmetz and C. Engler, "Human Perception of Media Synchronization," *Technical Report 43.9310*, IBM European Networking Center Heidelberg, Heidelberg, Germany, 1993.
89. R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications, and Applications*, Prentice Hall, Upper Saddle River, NJ, 1995.
90. C. Stoker, "From Antarctica to Space: Use of Telepresence and Virtual Reality in Control of a Remote Underwater Vehicle," *Mobile Robots IX, Proceedings of the SPIE*, vol. 2352, 1994, pp. 288–299.
91. D. Stone and K. Jeffay, "An Empirical Study of Delay Jitter Management Policies," To appear in *Multimedia Systems*.
92. D. Sullivan, D. Allan, D. Howe and F. Walls, "Characterization of Clocks and Oscillators," *NIST Technical Note 1337*, March 1990.
93. G. Tharp and S. Hayati, "Virtual Window Telepresence System for Telerobotic Inspection," *Telepresence Technologies, Proceedings of the SPIE*, vol. 2351, 1994, pp. 366–373.
94. B. Tseng and D. Anastassiou, "Compatible Video Coding of Stereoscopic Sequences Using MPEG-2's Scalability and Interlace Structures," *International Workshop on HDTV*, 1994.
95. D. Tsichritzis, S. Gibbs, and L. Dami, "Active Media," *Object Composition*, Dennis Tsichritzis Ed., Universite de Geneve, Centre Universitaire d'Informatique, Geneva, June 1991, pp. 115–132.
96. T. Wahl and K. Rothermel, "Representing Time in Multimedia Systems," *Proceedings of the International Conference on Multimedia*, Computing, and Systems, May 1994, pp. 538–543.
97. J. Webb, T. Warfel, and S. Kang, "A Scalable Video Rate Camera Interface," *Technical Report CMU-CS-94-192*, Carnegie Mellon University, Sept. 1994.
98. C. Wickens, "Three-Dimensional Stereoscopic Display Implementation: Guidelines derived from Human Visual Capabilities," *Stereoscopic Displays and Applications, Proceedings of the SPIE*, vol. 1256, 1990, pp. 2–11.
99. R. Yavatkar and K. Lakshman, "Communication Support for Distributed Collaborative Applications," *Multimedia Systems*, 2(2), 1994, pp. 74–88.
100. H. Zhang and E. Knightly, "A New Approach to Support Delay-Sensitive VBR Video in a Packet-Switched Network," *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, April 1995.
101. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, 7(5), September 1993, pp. 8–18.
102. L. Zhang, D. Estrin, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification," *Internet Draft*, Audio-Video Transport Working Group, file draft-ietf-rsvp-spec-07.ps, July 1995.