System Support for Efficient Dynamically-Configurable Multi-Party Interactive Multimedia Applications

Mark Moran[1] and Riccardo Gusella[2]

[1] The Tenet Group
Computer Science Division, University of California, Berkeley
and International Computer Science Institute
Berkeley, CA 94720

[2] Hewlett-Packard Laboratories
1501 Page Mill Rd.
Palo Alto, CA 94304

Abstract. An important new class of communication-oriented applications are described as multi-party interactive multimedia applications. In this paper, we examine the programming abstractions, system support and communication services required by these applications; evaluate existing support; and propose strategies for improving support by combining current systems with new designs. We have taken a holistic approach to this problem by considering the local system and communication architecture together in deciding where to implement each service, and in designing interfaces between components.

1 Introduction

Continuing advances in processing power, display capability, and communication capacity are enabling a class of demanding, communication-oriented applications that allow two or more geographically separated persons to interact as if they were at the same location. Escalating time and travel costs and the increasing need, in global markets, of personal and group communication, make these applications highly desirable. Applications that facilitate person-to-person collaboration by providing a variety of communication media—including video, voice, and shared drawing spaces—and support for shared applications—such as a spreadsheet, word processor, or designing tools—are classified as multi-party interactive multimedia (MIM) applications.

Current computer systems and communication architectures do not support MIM applications effectively for two major reasons: they do not provide abstractions and capabilities for MIM communication; and the device access, data copying, and data transport capabilities they do provide are inefficient and could potentially limit both the affordability and scalability of MIM applications. In this paper, we propose a holistic system approach to supporting MIM applications. The local computer system and the communication protocol stack are considered together in order to determine the best place to provide various services for MIM applications and to design interfaces among them. In Section 2 we discuss the requirements of MIM applications. Section 3 analyzes the strengths and weaknesses of current systems in meeting these requirements. Finally, in Section 4 we propose strategies for improving support for MIM applications by designing new abstractions and services to be combined with existing ones.

2 Requirements for Supporting MIM Applications

MIM applications require three types of system and communication support: programming and data abstractions that facilitate application development, a so-called real-time communication service for transmission of time-critical media, and efficient access to supporting services and multimedia I/O devices.

2.1 Programming abstractions

Development of multi-party interactive multimedia applications requires a richer set of programming abstractions and corresponding system support than traditional text- or graphics-based applications. Since MIM applications are inherently distributed, they must be able to operate across heterogeneous computing and communication environments. Therefore, abstractions must be provided to allow the flexible use of a variety of multimedia I/O devices and real-time communication services. We have identified the following abstractions as being fundamental:

Time-based media. Like general multimedia applications, MIM applications require support for time-based media, including device-independent functions to handle the acquisition, presentation, and transmission of time-based media. Presentation functions may perform synchronization between multiple streams.

Multi-party streams. The model for a multi-party stream that we will use is a media stream with one source and multiple destinations. It is important that the source be able to control and *transmit* such a stream without being concerned about the (possibly dynamic) set of participants currently receiving the stream. An abstraction of multi-party streams simplifies programming and enables abstract, efficient access to underlying multicast capabilities.

Receiver control over local presentation devices. Each human user must have the ability to control access to his or her local presentation devices, including the ability to suspend and resume any media stream. Users should be able to determine how media from various streams are multiplexed on an I/O device, and to influence the tradeoffs made between the quality of presentation and the cost of transmission and processing for each stream.

Security abstractions that can be used to protect participants from unwanted eavesdroppers and disrupters are also important. However, we will not consider these abstractions in this paper.

2.2 Communication requirements

The communication requirements of a MIM application are two-fold. First, the communication service must meet certain performance requirements of the application, which we shall denote as Quality of Service (QoS) requirements. The second communication requirement, one of efficiency, is for a multicast service.

Real-time communication service. Because MIM applications support real-time interactions between human users using time-based interfaces, they require a communication service that maintains the QoS requirements designated at the beginning of the conversation, regardless of fluctuations in network load. Without such a service, users may experience difficulty interacting and may notice gaps in the presentation of time-based media. A real-time communication service may offer either *hard guarantees* that have a strict analytical basis (e.g. [14]), or *soft guarantees* that indicate a target quality of service, but lack firm guarantees (e.g. [6]). A real-time communication service must include at least the following elements:

- *Service interface.* The service interface allows a potential network client to describe its expected traffic and to specify its QoS requirements. The expressiveness of the service interface determines the accuracy with which an application may describe its traffic and the quality of service it may request.
- *Support at network switching nodes for guaranteeing QoS.* At the very least, network nodes must be able to distinguish between traffic requiring real-time support, and unprotected non real-time traffic.
- *Admission control.* The network must refuse to accept new conversations when it determines that either (a) the network will (may) not be able to meet the QoS requirements of the new conversation, or (b) that accepting the new conversation will (may) cause the QoS requirements of existing conversations to be violated.
- *Protocol support.* Protocol support consists of the communication protocols that are required to implement the QoS support mechanisms of the internal network, to gather information required for admission control decisions, to establish and manage conversations, etc. Network protocols that actually suspended transmission of data to multicast receivers that have suspended a stream, would enable human users to conserve network and system resources by suspending streams.

Multicast ordering and reliability guarantees. Because most multicast ordering guarantees are incompatible with bounded delays, and because time-based media is usually "stateless", ordering and reliability guarantees are of limited usefulness for the transmission of time-based media. However, ordering and reliability guarantees are useful for MIM communication involving shared workspaces, such as a distributed drawing space or a shared application that allows input from multiple users; and for implementing *floor-control* algorithms that determine which participant of a multi-party conversation is enabled to transmit video and audio stream to the other participants. (e.g. [19]).

2.3 Efficiency requirements

Although some may claim that improvements in processor, memory and network speeds will alleviate efficiency concerns in the future, it seems very likely that efficiency will remain an important concern for MIM applications, since efficiency directly affects the quality, size, and number of simultaneous conversations in which a user may participate. Certainly, in the near future efficient utilization of system and network resources is crucial to ensure the affordability of MIM communication.

Efficiency concerns fall into two major categories: optimization of services, and efficient coordination between services. Optimizations of services includes efficient implementations of a service (e.g. multicast), efficient data handling (i.e. reduction of copying), and dynamic access to available hardware-assist modules, (e.g. a decompression module). Efficient coordination between services includes issues such as eliminating duplication of services between layers and designing interfaces that allow higher layers to access lower-level services efficiently. A holistic system approach is important for improving efficiency.

3 MIM Support in Existing Systems

A number of current systems provide partial support for MIM communication. In this section we will describe a few representative examples and analyze how well each meets the requirements specified in the previous section. We have grouped these systems into those that provide programming and system abstractions and those that provide communication services; efficiency considerations will be addressed in each section. After describing each system, we list its advantages (+) and disadvantages (–) for MIM applications, using **boldface** to indicate a reference to the requirements described in the previous section.

3.1 Programming and System Abstractions

Current systems provide programming abstractions and support for handling time-based media either by extending the client-server paradigm of a window system to handle time-based media and I/O devices (e.g. [1]), or by extending the operating system (e.g. [16] [2]).

Window Systems. Window systems vary considerably in their event-handling architectures, support for graphics, and programming philosophy. However, the features that are of primary importance to MIM applications—abstractions for network communication and for dynamic access to heterogeneous I/O devices—are relatively similar. Therefore, we will consider the X window system [21] as a representative example in discussing the support for MIM applications provided by network-transparent window systems.

The X window system has a client-server architecture: client applications (so-called *X clients*) communicate via a machine-independent protocol with a user-level process (the so-called *X server*) that multiplexes I/O to and from I/O devices (video display, keyboard,...). The use of a machine-independent protocol between client and server allows an application to access a remote display *transparently* without including explicit networking code.

Dynamic access to heterogeneous I/O devices is handled in two ways: device-independent abstractions are defined for accessing I/O devices, and a rudimentary query capability allows clients to determine and adapt to the characteristics of a remote video display. The *Xv* extension provides some support for incorporating *analog* video by providing access to devices that digitize analog video into (or produce analog output from) the framebuffer. Digitized frames may be grabbed out of the framebuffer.

+ Window systems provide an architecture that allows **dynamic, device-independent access to I/O devices** and abstractions for networking.
+ The Xv extensions allow **device independent** access to **video adapters** that enables display and export of analog video signals.
– One important weakness in current window systems is a complete lack of support for **digital time-based media**.
– Another important short-coming is that the networking abstraction does not support access to **multicast** or **real-time communication services.** Just as important, the networking abstraction cannot describe **dynamic**, participation in **multi-party communication.**
– Xv has been designed to support local display of video, not **distribution**. This decision results in two problems: because the signal is digitized directly into the framebuffer, analog video must be displayed locally in order to be distributed; and access to digitized frames through the X server is inefficient.

Although standard window systems do not provide much support for MIM applications, their philosophy of providing abstractions for networking and device I/O can be extended to support time-based media, either by extending the X server itself or by introducing a companion I/O server for time-based media. The next two systems demonstrate these two strategies.

Extending the Window System: SharedX. SharedX[3] is a commercial product that allows windows of existing applications to be shared by multiple, separated users. The application to be shared first establishes a standard X connection with the *SharedX server* (a modified X server). Through a graphical interface provided by the *SharedX client*, the user indicates a list of destination displays for each window that will be shared. The SharedX server distributes X protocol messages regarding shared windows to the remote X servers as if it

[3] SharedX is a registered trademark of Hewlett Packard Co.

were an X client. Neither the client application nor the remote X servers need to be changed.

+ The SharedX server provides limited **multi-party communication** between an X client and remote X servers.
+ The SharedX client provides abstractions and user-interfaces for describing **multi-party communication**; thereby simplifying application development and ensuring uniformity of abstractions and interfaces.
– As with X windows, no support is provided for **time-based media**.
– Also as with X windows, the communication abstraction does not afford access to underlying **multicast** or **real-time communication services,** or **dynamic** participation in **multi-party** communication.
– The granularity of **receiver control** is too coarse: in order to share a window, a remote machine must give the machine running the SharedX server complete access to its display.

Separate I/O Server for Time-Based Media: ACME. Anderson, Govindan and Homsey have designed and implemented the ACME I/O server that provides abstractions for device-independent access to continuous media (time-based) I/O devices, and a *logical time system* abstraction that manages timing of, and synchronization between, continuous media (CM) streams [1]. Nicolaou has proposed a similar design in [18]. Govindan and Anderson have also designed and implemented extensions to the UNIX[4] operating system that provide efficient multi-level process scheduling, and *memory-mapped streams* that reduce data copying in order to support efficient implementation of the ACME server as a user-level process [15].

+ Good support for **time-based media**: access to **continuous media I/O devices** and mechanisms for handling the **timing and synchronization** of CM streams.
+ Operating system extensions to make implementation **efficient**.
– ACME supports neither access to **underlying multicast** capabilities nor the description of **dynamic** participation in **multi-party** communication.
– There is no notion of **variable quality,** hence no capability for users to trade off quality against resource utilization.
– Querying capabilities that would allow **flexible access** to I/O devices and hardware processing modules (e.g. video codec) are not provided.
– It does not seem possible to describe and utilize specialized data paths in a machine-independent manner, e.g. a direct data path between a decompression module and a frame buffer.

Operating System Extensions for Time-Based Media: QuickTime. The QuickTime extensions to the Apple Macintosh operating system and associated

[4] UNIX is a trademark of USL

toolbox routines support creation, manipulation and display of time-based media as represented by a *movie* abstraction. The movie abstraction associates and describes related audio and video streams, including timing information for playback. It is designed to support fast cut-and-paste editing and efficient sharing of time-based media data between movies. Toolbox support offers flexible and dynamic access to frame grabbers, and hardware or software compression modules.

+ Excellent programming support for generation, manipulation and playback of stored **time-based media**.
+ **Dynamic, device-independent access** to I/O devices and hardware modules that capture and present time-based media.
+ Some support for **user-selectable quality** on playback.
– The current abstractions do not support **network communication** between distributed systems. Although networking capabilities have been announced for the next version of QuickTime, the overall QuickTime paradigm and past Apple networking abstractions suggest that networking will not provide access to **multicast** or **realtime communication** services.
– The *movie* abstraction is not well-suited to describe **distribution** of time-based media, particularly **interactive, multi-party communication.** The authors believe that interactive communication is more naturally represented as a flow of data between devices than as production and immediate playback of a movie to and from a "network device". For example, the *record-playback* paradigm of a movie suggests the use of large buffers in production to prevent loss of data; while interactive applications would prefer to suffer momentary data loss in order to decrease end-to-end delay. An indication of the bias towards a record-playback paradigm is the movie file format, in which the movie *header,* which includes the length of the movie, appears at the end of the file, unnecessarily complicating network transmission.

3.2 Network Communication Services

In this section we describe three proposals to provide communication services for the transmission of time-based media. Because of space considerations, we have selected these three as representatives of a rich area of research. We also describe the IP multicast extensions.

ISI/BBN Protocol Suite. ISI and BBN have collaborated to develop a suite of communication protocols that support N-way video conferencing [5]. The network protocol is ST-II, which provides a connection-oriented service that includes multicast and QoS guarantees [4]. At the transport-level, NVP [7] and PVP [8] support transmission of voice and video streams respectively. The Connection Control Protocol (CCP) [22] manages connections and performs synchronization between streams.

+ These protocols have been used successfully to support video-conferencing across the DARPA Terrestrial Wideband Network [11].
+ ST-II provides 1 $\Rightarrow$ N **multicast**, **dynamic addition** and **deletion** of multicast receivers, and admission control in order to meet **QoS requirements** for the transmission of time-based media.
– Transport-level protocols are *special purpose* for each media type (hence a separate protocol for voice and video). They provide no meaningful **abstractions for time-based media** in general.
– Currently, **performance guarantees** are supported in the Terrestrial Wideband Network via *ad hoc* rather than general algorithms. These techniques do not seem applicable to other networks. Allocation of peak rate bandwidth may artificially reduce the *proportion* of network resources that can be utilized by *real-time* clients. (System resources that are unused by real-time traffic will be available to non real-time traffic.)
– No **receiver control** over transmission of time-based media streams.

Tenet Real-Time Protocol Suite. The Tenet real-time protocol suite provides deterministic and probabilistic guarantees for real-time communication [14]. Analytical algorithms based on worst-case assumptions are used to provide deterministic and probabilistic guarantees. A network-level protocol, RTIP [23], provides simplex, unicast connections with a guaranteed lower bound on bandwidth, and guaranteed upper bounds on end-to-end delay, delay jitter, and buffer overflows within the network. CMTP [17] is a transport-level protocol that provides abstractions to simplify and improve efficiency for the transmission of time-based media. Admission control and resource allocation for both protocols is handled by the Real-Time Channel Administration Protocol (RCAP) [3]. RCAP has a hierarchical architecture that enables it to perform admission control across internetworks of arbitrary topology.

+ RTIP provides flexible **real-time service** to support transmission of time-critical data (including time-based media) according to **QoS requirements** specified by the **application**.
+ Analytical basis of algorithms provides predictable performance that greatly simplifies application development. It also allows admission-control decisions to be made in a *distributed* manner, without requiring global knowledge of network state.
+ RCAP implements admission control algorithms via a hierarchical approach that allows the scheme to work across **heterogeneous internetworks** of **arbitrary topology**.
+ CMTP provides **more efficient** service and **better abstractions** of communication for time-based media than traditional message oriented paradigms [13]. The use of a shared buffer between the sender and the transport service allows data transmission to proceed *periodically* after the start of a stream without further synchronous interactions with the sending applications.
– No **multicast communication.**

– No support for **receiver control** or manipulation of streams.
– Worst-case analysis may artificially reduce the *proportion* of network resources utilized by *real-time* clients. Unused resources may be utilized by non real-time traffic.

Predicted Service. An alternative proposed by Clark et. al is *predicted* service [6]. As with the Tenet suite, a client application describes its traffic and a requested delay bound during an establishment phase. However, instead of basing admission control decisions on analytical models, the current load is measured and used as a predictor of future network load. Predicted service is designed for *adaptive* applications, which try to reduce end-to-end delays (*playback point*) as network load decreases, and which adapt to increases in delays as load increases.

+ Traffic description and **QoS requirements** are **flexible** and can be adapted to **application requirements.**
+ Emphasis on measurement and statistical multiplexing will probably allow a higher-proportion of network resources to be used for *transmission of time-based media* than worst-case analysis.
– Variation in the end-to-end delay *bounds* for a stream, caused by the inaccuracy of using current load as a predictor of future load, requires applications to modify their behavior and may introduce glitches in the display of time-based streams.
– No support for **multicast.**
– No support for **receiver control** for the suspension or manipulation of time-based media streams.

IP Multicast. Deering has designed and implemented extensions to IP to support multicast delivery of IP datagrams to dynamic distribution lists [10]. The assumed network topology is a graph of IP multicast routers connecting broadcast-based leaf networks. Part of the IP address space is designated as logical multicast addresses. Multicast delivery is implemented in a two-level hierarchy: multicast routers distribute messages among themselves and broadcast them to each attached network on which a host is receiving the multicast group. Hosts filter broadcast packets based on IP addresses..

+ **Multicast** service is provided with relatively little additional overhead or modifications to existing protocols.
+ The protocol is virtually stateless: a recovering multicast router is able to fully reconstruct its state by querying attached networks.
+ Hierarchical approach gives reasonable **scalability.**
– The most important weakness is that no provision is made for **real-time performance guarantees.**
– Selection of transient multicast IDs allows for collisions and unpredictable disruptions of service.

- Little or no **receiver control** over transmission of streams. In particular, the time to disconnect from a multicast stream is non-deterministic, causing host resources to continue to be consumed for a longer time than necessary.

4 Strategies for Supporting MIM Applications

We have developed strategies that combine existing systems with new approaches to support MIM communication efficiently and flexibly. An important characteristic of our proposal is a holistic approach to supporting MIM communication. Rather than focusing on a single layer or service, we have considered the entire system and communication architectures together, in order to determine the best place to implement each service, and the proper interfaces between layers to allow abstract, efficient access to such services. The following subsections describe the strategies we have developed for providing programming abstractions and system support, and abstractions of communication services.

4.1 Programming Abstractions and Local System Support

As an overall design architecture, we have decided to provide programming abstractions and local system support through the implementation of a continuous media I/O server co-resident with a standard window server, as in ACME. Our reasons for choosing this approach over media-specific extensions to the operating system are as follows:

1. Applications based on a portable I/O server will themselves be more portable and will be easier to distribute across heterogeneous platforms. This point is demonstrated by current window systems.
2. Applications based on a portable I/O server can run on existing operating systems and can be migrated more easily to future systems that include extensions for time-based media and multi-party communication.
3. By separating the data and control paths, a user-level I/O server can utilize optimized I/O routines and data paths in the operating system and can, therefore, match the performance of operating system extensions.
4. A user-level I/O server will be more flexible and more easily extended than an operating system.

Elements Drawn from Existing Systems. Elements drawn from ACME, QuickTime, and Xv fall into three categories: device-independent access to time-based I/O devices, querying capabilities to allow dynamic utilization of available devices and modules, and support for timing and synchronization of time-based media streams.

Device-Independent Access to I/O Devices. In general, devices are classified as either sources or sinks of time-based media. Routines are provided to move data to and from such devices. As with QuickTime, access to modules that process CM data (e.g. compression/decompression modules) is also supported.

Querying capabilities. Querying facilities (provided in QuickTime, X, or Xv), are required to allow distributed applications to determine the I/O devices and processing modules available at a remote site for processing and presenting time-based media. Since we have chosen an architecture similar to current window systems, we propose query routines that accept a template of the desired device or module.

Timing and Synchronization of Time-Based Media. The *playback* and *capture* routines that provide device-independent access to devices, will also handle the timing of media capture and presentation. An application accessing such devices will need only to initiate and terminate capture or presentation of the stream. Both ACME and QuickTime also provide support for synchronizing the presentation and, in the case of ACME, the capture of related CM streams through the use of logical time systems.

New Designs. In order to support MIM applications, we have developed new abstractions for efficient, device-independent data handling and a richer encapsulation of network communication that enables abstract access to real-time, multicast communication services.

Abstractions for Data Handling. Because of the relatively high data rates required to support MIM communication, we have developed an abstraction for data movement that allows efficient data handling without introducing machine dependencies in the application. The primary strategy of our design is to separate control of data handling from the data path itself by using a *setup* phase. During the setup phase, the application indicates the flow of data between *logical* devices, where a *logical device* may correspond to a physical device (e.g. a framebuffer), or a software module (e.g. an I/O server process that maintains synchronization between several media streams). Data is then moved between logical devices without requiring further intervention by the application. This approach can take advantage of specialized data paths, (e.g. between a video decompression module and the framebuffer) and/or optimized data handling provided by the operating system (e.g. as described in [12]), without requiring modifications to the application.

The interface we have designed uses the abstraction that every device or module describes the location of its own *input buffer*, which may be in system memory or on a physical device. This description is passed as a parameter to the *preceding* logical device during the setup phase. The preceding device is then responsible for directing its output in the most efficient manner into the input buffer of the logical device following it. This abstraction also allows data paths to be altered dynamically, e.g. to switch between input devices.

Abstractions for Network Communication. We have also designed richer networking abstractions to describe dynamic, multi-party communication and to allow system-independent, efficient access to multicast and real-time communication services. This design consists of three components: a logical display to describe multi-party communication; an abstract interface for a real-time communication service; and connection management mechanisms.

- **Logical display.** The networking abstraction supported by the X window system consists of a *display* structure that specifies a connection between a client application and the destination X server. In order to support multi-party communication, we have extended the display structure to a *logical* display that keeps track of participants and destination addresses, and supports dynamic addition, deletion, and suspension of participants.
- **Abstract service interface to real-time communication.** Instead of trying to hide network access, we have developed an abstraction of network communication that enables simple and efficient access to communication services supporting real-time communication and multicast. Access to the network is managed by the I/O server, just as other I/O devices. One advantage to managing network communication in the I/O server is that the same networking abstractions can be applied to a wide variety of networks and internetworks, with the I/O server performing translation of parameters as required.

  Applications describe their communication requirements in terms of a high-level model of multi-party communication that can be mapped precisely onto the network-level models described in [14], [17], [20], [9], and [6].
- **Connection-management mechanisms.** We have also developed connection management abstractions that allow users to join, leave, suspend and resume connections. These capabilities allow a (human) receiver to determine interactively how local system and network resources are proportioned between streams.

4.2 Network Communication Services

The strategies we have developed for supporting the communication requirements of MIM applications combine the real-time communication services and multicast abilities of existing systems with new design features to support scalability by promoting sharing of network resources between related streams.

Elements Drawn from Existing Systems. Real-time communication services will be based upon extensions of, or modifications to, existing systems described in Section 3.2. In addition, we will include the stream-oriented paradigm of time-based communication presented in CMTP and the dynamic membership of multicast groups supported by both ST-II and IP multicast.

New designs. We have targeted two areas for future research in communications support for MIM applications: general algorithms for real-time multicast services, and mechanisms for promoting the sharing of network resources.

Dynamic, Real-Time Multicast. In order to extend a unicast real-time service to support multicast, we are defining new models for traffic characterization and QoS requirements and exploring new routing and resource allocation schemes. Incremental routing mechanisms are being studied in order to support dynamic additions, deletions, and possibly suspensions of participants.

Mechanisms to promote sharing. The real-time communication services we have described in Section 3.2 have each made one of two assumptions about the relationship between streams. Services that provide analytical, worst-case guarantees assume that all streams are independent (e.g. [14]). Other services attempt to promote sharing of network resources by assuming (i) that current measured network behavior is a good predictor of future behavior; and (ii) that statistical multiplexing between streams will mask the burstiness of any single stream (e.g. [6]). However, we propose sharing resources between related streams according to a priori knowledge of the relationship between the streams. For example, a MIM application may use a *floor control* mechanism to serialize audio and video input from participants. Network resources can then be shared between video and audio streams from different sources, even if statistical multiplexing does not apply, e.g., in the host or over any link on which one stream uses a significant portion of network resources.

Conclusion

We have described system and communication support for multi-party interactive multimedia (MIM) applications. In particular, we have shown that a holistic approach is important to ensure that required services are provided at the appropriate place in the system and/or communication hierarchy and to define interfaces that enable abstract, efficient access to underlying systems. We have examined existing systems and proposed several innovations to improve the support given to MIM applications. In particular, we have designed abstractions that enable abstract, efficient access to real-time multicast communication services, multimedia I/O devices and support services.

Acknowledgments

The authors would like to thank many colleagues at Hewlett Packard Labs and the Tenet group for their invaluable discussions, advice, and comments. Walt Hill, Tracy Steelhammer, Srinivas Ramanathan, Audrey Ishizaki, Amit Gupta, Clemens Szyperski, and Giorgio Ventre have been of particular assistance.

References

1. D. P. Anderson and G. Homsy. A continuous media I/O server and its synchronization mechanism. *IEEE Computer*, 24(10):51–57, Oct. 1991.

2. Apple Computer Inc. QuickTime version 1.0 for developers. CD-ROM documentation, 1991.
3. A. Banerjea and B. Mah. The Real-Time Channel Administration Protocol. In *Proc. 2nd Int. Workshop on Netw. & O. S. Support for Dig. Audio and Video*, Nov. 1991.
4. S. Casner, J. Charles Lynn, P. Park, K. Schroder, and C. Topolcic. Experimental internet stream protocol, version 2 (ST-II). *RFC 1190*, Oct. 1990.
5. S. Casner, K. Seo, W. Edmond, and C. Topolcic. N-way conferencing with packet video. In *Proc. 3rd Int. Workshop on Packet Video, VISICOM '90*, Mar. 1990.
6. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Arch. and mech. In *Proc. SIGCOMM '92*, 1992.
7. D. Cohen. A network voice protocol NVP-II. Technical report, USC/Information Sciences Institute, Apr. 1981.
8. E. Cole. PVP - a packet video protocol. Technical report, USC/Information Sciences Institute, Aug. 1981.
9. R. L. Cruz. *A Calculus for Network Delay and a Note on Topologies of Interconnection Networks.* PhD thesis, University of Illinois, July 1987.
10. S. E. Deering. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, Dec. 1991.
11. W. Edmond, K. Seo, M. Leib, and C. Topolcic. The DARPA wideband network dual bus protocol. In *Proc. SIGCOMM '90*, pages 79–89, Sept. 1990.
12. K. Fall and J. Pasquale. Exploiting in-kernel data paths to improve I/O throughput and CPU availability. In *Usenix Winter Conference*, 1993.
13. D. Ferrari, A. Gupta, M. Moran, and B. Wolfinger. A continuous media communication service and its implementation. In *Proc. of GLOBECOM '92*, 1992.
14. D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE J. on Sel. Areas of Commun.*, Apr. 1990.
15. R. Govindan and D. P. Anderson. Scheduling and IPC mechanisms for continuous media. In *Proc. of Symp. on O. S. Principles.*, pages 68–80, Oct. 1991.
16. E. Hoffert et al. QuickTime: an extensible standard for digital multimedia. In *Proceedings of CompCon '92*, pages 15–20, Feb. 1992.
17. M. Moran and B. Wolfinger. Design of a continuous media data transport service and protocol. Technical Report TR-92-019, Int. Computer Science Inst., 1992.
18. C. Nicolau. An architecture for real-time multimedia communication systems. *IEEE Journal on Selected Areas of Communications*, 8(3):391–400, Apr. 1990.
19. T. Ohmori et al. Multiparty and multimedia desktop conferencing system: MERMAID. In *Proc. of IEEE Multimedia '92*, pages 122–131, Apr. 1992.
20. C. Partridge. A proposed flow specification. Internet RFC 1363, Sept. 1992.
21. R. W. Scheifler and J. Gettys. The X window system. *ACM Transactions on Graphics*, 5(2):79–109, Apr. 1986.
22. E. Schooler and S. Casner. An architecture for multimedia connection management. In *Proc. of IEEE Multimedia '92 Workshop*, pages 271–274, Apr. 1992.
23. H. Zhang, D. Verma, and D. Ferrari. Design and impl. of the Real-Time Internet Prot. In *IEEE Wrkshp on the Arch. and Impl. of High Perf. Comm. Subsys*, 1992.