The Multimedia Internet Terminal (MINT)*

Dorgham Sisalem
GMD-Fokus, Berlin
sisalem@fokus.gmd.de

Henning Schulzrinne
Columbia University, New York
schulzrinne@cs.columbia.edu

Abstract

The Multimedia Internet Terminal (MINT)[1] is a flexible multimedia tool set that allows the establishment and control of multimedia sessions across the Internet. The system architecture is fully distributed, with no central components. For each participant, a coordinating application manages a set of loosely coupled media and control agents. Using the coordinating application, session members can control the sending and receiving of several audio and video streams, invite other users to the session and negotiate the order in which members are allowed speak. The different components of MINT are separate processes that are independent from one another, yet can easily communicate with each other using a simple interprocess communication protocol based on local multicast. Individual components can be replaced without affecting the operation of other components. The same media agents can be used in applications ranging from media-on-demand retrieval to Internet telephony and distance learning, simply by using different control tools.

To improve the quality of service of the multimedia sessions MINT is enhanced with a suite of QoS control mechanisms including resource reservation based on RSVP, adaptive media control and the ability to send video data in hierarchically layered streams. Thus, the tool set can accommodate a range of capabilities and available network bandwidths in heterogeneous environments.

KEY WORDS: *Audio/video conferencing; floor control; session initiation; Session Initiation Protocol (SIP); Session Description Protocol (SDP); Resource Reservation Protocol (RSVP); adaptive applications; intermedia synchronization; signaling; QoS control.*

1 Introduction

The rapid growth, increasing bandwidth and the availability of low-cost multimedia end systems has made it possible to use the Internet for multimedia applications ranging from

*This work was funded in part by the BMBF (German Ministry of Education and Research) and the DFN (German Research Network) under the USMInT project.

[1] MINT is available free software from `http://www.fokus.gmd.de/step/mint`

telephony to conferencing, distance learning, media-on-demand and broadcast applications [30]. However, using the Internet for audio/visual communication introduces a range of problems that were not addressed in the original design of the network, its protocols and applications. The design of the Multimedia Internet Terminal (MINT) described here addresses some of these issues and incorporates some of the most recent proposals for solving these problems. Handley et al. [13] summarizes some of the main issues related to the subject of multimedia communication over the Internet. Below, we describe how MINT addresses these issues. As will be discussed in more detail later, we distinguish media agents that process continuous media streams from control agents that initialize, coordinate and control these agents.

Intermedia interaction: In a networked multimedia application, media agents and controllers have to interact on each end system so that media can be lip-synched, the generation and reception of media streams can be controlled via floor control and new services can be implemented. The different agents constituting MINT can communicate with each other using a simple control protocol, based on local multicast, called Pattern Matching Multicast (PMM) [27]. This gives the user the impression of having a monolithic tool, that consists, however, of separate components that can be easily replaced, upgraded or adapted to new applications.

User location and invitation: To initiate a multimedia session, all participants need to know a set of parameters such as the session (multicast) address and the set of media, their port numbers and their encodings. Prospective participants can either discover sessions on their own or be explicitly invited. Participant-directed discovery includes listening to session announcements being distributed via mechanisms such as a well-known multicast address [12] or web pages with listings of events and on-demand multimedia streams. Announcements via multicast or email lists are only appropriate for a relatively small number of pre-scheduled public sessions of wide interest. Users could be invited explicitly via email, but the delay in delivery and reading makes telephony-like spontaneous or private conferences difficult. Recently, several proposals for sending invitations for multimedia sessions were developed [28, 15]. By locating the actual end systems the users are logged on and informing them about the session, the task of initiating multimedia sessions becomes much easier and resembles to some extent the old but reliable telephone system. MINT includes the latest version of the session initiation protocol (SIP) [15]. Thereby, users of MINT can invite other users in a simple way to join multimedia conferences. After accepting an invitation MINT initiates the needed startup procedures on their behalf, thereby minimizing the complexity of using the system.

Conference control: To simplify the task of controlling and using a multimedia conference, MINT provides each user with a conference control agent that controls all media agents the member is using in the session. The conference control agent offers a single, consistent user interface regardless of the type of media agent used. This is in contrast to current Internet multicast tools, where simple tasks like discovering which video image belongs to the current speaker are tedious. Our approach also reduces the number of windows the member needs to monitor and position. In addition to the audio and video agents, MINT includes agents for floor control and quality-of-service (QoS) control.

QoS control: The Internet currently supports only best-effort service. To allow for an improved QoS, two different approaches are currently being discussed. The integrated service model [3] has been proposed within the Internet Engineering Task Force (IETF), the standardization body of the Internet. It allows end systems to reserve the amount of resources needed to fulfill their requirements. On the other hand, various adaptation schemes [2, 5, 23] are being discussed that adjust the end system requirements to the current congestion state of the network. MINT can benefit from both approaches, as it implements the resource reservation (RSVP) protocol for establishing guaranteed QoS reservations and supports several adaptation schemes in the video agents.

The paper is structured as follows. In the next section (Section 2), we describe related work in the area of multimedia conferencing. In Section 3, we describe the architecture of the Multimedia Internet Terminal (MINT) in detail, as well as the protocol used for the communication between the different agents. In Section 4, we describe the different parts of the MINT. Currently, MINT consists of a video media agent (NeViT) [36], an audio media agent (NEVOT) [26], a session controller (ISC), and invitation, reservation and floor control agents.

Section 5 then describes different approaches for providing quality of service, namely resource reservation, rate adaptation and hierarchical data transmission. Finally, in Section 6 we conclude the paper and present some of the issues we are still working on.

2 Background and Related Work

Two main issues need to be considered when realizing multimedia conferencing over the Internet: the network protocols used and the available applications. In this section, we briefly describe some of the different characteristics of Internet multimedia applications and networking issues essential to realizing multimedia conferences and other applications.

2.1 Network Support for Multimedia Conferencing

Our work focuses primarily on the delivery of packetized multimedia in the Internet, often distributed using IP multicast. Currently, IP multicast is deployed in the Internet as an overlay network referred to as the MBone [6, 9]. However, most local networks also support multicast. Following convention, we refer to tools operating in this environment as *MBone tools*, although they all can use any multicast-capable IP network and all can operate in unicast mode. The protocol stack for Internet multimedia is shown in Fig. 1.

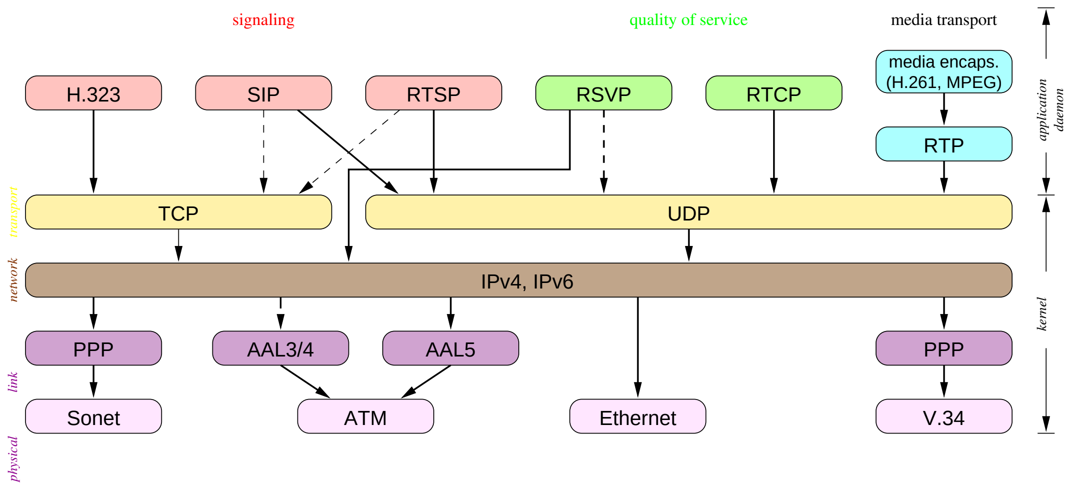


Figure 1: Internet multimedia protocol stack

Most of the current available MBone tools such as VIC[22], VAT, rat [20], or NEVOT[26] use the real time transport protocol (RTP) [31] designed within the Internet Engineering Task Force (IETF). RTP is now widely used for conferencing, as well as emerging applications such as Internet telephony and Internet media-on-demand services. It is also part of ITU recommendation H.225.0 for packet-based conferencing.[2]

RTP is an end-to-end protocol that is often used together with other transport protocols, in particular UDP. RTP sessions consist of two lower-layer data streams, namely a stream of media data such as audio or video and a stream of control packets that is carried a sub-protocol called RTCP. The data protocol offers functionality common to a range of media types, including timing, loss detection, resequencing, payload type marking, encryption and demarcation of application layer data units such as video frames and voice talk-spurts. Each RTP packet also contains a random identifier that makes it possible to distinguish several sources that traverse a single transport-level "translator" such as a firewall. A packet may also indicate a list of "contributors", useful for maintaining identities when mixing audio streams or playing back recorded RTP sessions.

RTP has no notion of a connection; it may operate over either connection-oriented protocols such as ATM AAL5 or connectionless lower-layer protocols, typically UDP/IP. It does not depend on particular address formats and only requires that framing and segmentation is taken care of by lower layers. RTP offers no reliability mechanisms. It is typically implemented as part of the application or as a library rather than integrated into the operating system kernel.

In UDP, data and control streams use separate ports, however, they may be packed into a single lower-layer stream as long as RTCP packets precede the data packet within the lower-layer frame. A single stream may be advantageous for systems where connections may be costly to manage, e.g., ATM PVCs [3].

The control protocol (RTCP) allows monitoring of the received and transmitted data rates, delay jitter and packet losses. Each session member periodically multicasts control packets to all other session members containing information about the amount of data sent, if any, and reception reports for each sender. Each session member also includes a globally unique identifier and possibly other identifying information such as the member's name or email address. All participants share a fixed, constant control bandwidth, typically set at 5% of the data bandwidth, with data senders getting a disproportionate share. Control traffic is typically sent best-effort, but, since its bandwidth is small and known, it can be figured into the resource reservation. RTCP packets also contain identifying information that allow to connect different media streams with the same participant.

[2] http://www.cs.columbia.edu/~hgs/rtp contains a listing of RTP-capable tools.

[3] The video engine of MINT currently uses two separate VCs for data and control packets when transmitting over native ATM connections.

2.2 Architecture of Multimedia Conferencing Application

When looking at the currently available multimedia conferencing tools two different approaches can be distinguished:

- The first type of tool is a single, large, monolithic tool that support different tasks such as video, audio and application sharing. Such tools consist either of a single program or a tightly integrated set of applications that can only interoperate within the set. Adding new features to such tools, applying them to new tasks or upgrading the media agents by, say, replacing a video agent by a faster one is rather difficult.
- Secondly, loosely coupled tools such as the Internet multicast (MBone) tools listed earlier. In this approach, each media is typically handled by a distinct media agent. Conference control issues such as floor control, joining and leaving a conference and starting the appropriate agents is delegated to an external conference controller. With such an approach, the media agents can be easily replaced, updated and reused in new applications. The main drawback of this approach is that once the conference controller has started a media agent, this media agent is on its own and the controller is no longer in control of it. This means that the user has to employ a different user interface for each application, with all of the interfaces doing in part common tasks such as initiating or terminating a service or displaying the members of the session. A session member contributing audio and two video streams appears in three different places in the user interface, possibly with different name display conventions and relative positioning. A more severe drawback of the loosely coupled tools approach is the lack of interaction between the different tools. It is also difficult to build "embedded applications" that hide some of the underlying complexity from the user. For example, for an "Internet TV" application, listing all other viewers is hardly appropriate.

This rough characterization of multimedia tools indicates the basic design concepts a multimedia tool set should follow: different agents should easily interact with each other, yet there should be no dependencies between the different components. MINT was designed to fulfill these requirements. It consists of several applications that can communicate with each other using a simple communication protocol. The applications are, however, independent of each other so that they evolve independently.

Another distinguishing characteristic of multimedia applications is their approach to session control. *Light-weight* (or loosely-controlled) sessions are multicast based and lack explicit controls on session membership and centralized control of media sending and receiving. With this approach, conference control information is usually multicast to all members. Light-weight sessions can easily scale to several million participants, covering the whole spectrum from a two-party phone calls to Internet video broadcasts. Since any multicast-capable Internet host can subscribe to any multicast address, light-weight conferences have to rely on encryption to ensure privacy. Due to packet delays and losses, no single member can keep a complete and current list of all participants. Multicast groups are also subject to intentional and accidental denial of service problems. Most of the applications currently used in the MBone are based on this architecture; however, a set of tools using the H.323 suite of protocols follows the model of *tightly coupled conferences*. Tightly coupled conferences use explicit conference membership mechanisms and often have an explicit conference control mechanism regulating who can send data into the conference. They often use a hub-and-spoke model, with a multipoint control unit (MCU) serving as a rendezvous point and possibly replicating media streams from a sender to all participants. Conferencing applications used in ISDN environments also usually rely on this approach.

For small groups, tightly coupled conferences avoid the need for multicast, which is not yet widely available in the Internet. An MCU, if used, also offers access-based security, restricting who can readily listen or send media data to the conference. Billing is also simplified. However, tightly coupled conferences introduce difficult state synchronization problems, have single points of failure and may provide a false sense of security in the absence of strong encryption.

Since we want our set of tools to span the whole range of Internet multimedia applications, MINT follows the light-weight conference model.

3 Local Conference Control Architecture

The previous section discussed the flexibility of loosely coupled tools. However, this flexibility comes at the cost of complicating the interaction between the different applications used within the conference.

Intermedia synchronization, for example, requires that audio and video agents interact.

Audio and video streams arrive at the destination nodes out of synchronization due to the different delay jitter they experience within the network. The delay incurred by audio encoding and decoding also typically differs from that for video. Other examples include quality of service control, automatically displaying the video from those conference participants that are currently talking or using the display coordinates of video windows to control artificial spatial placement of the audio from individual speakers (artificial stereo, holophony) [7, 19, 21].

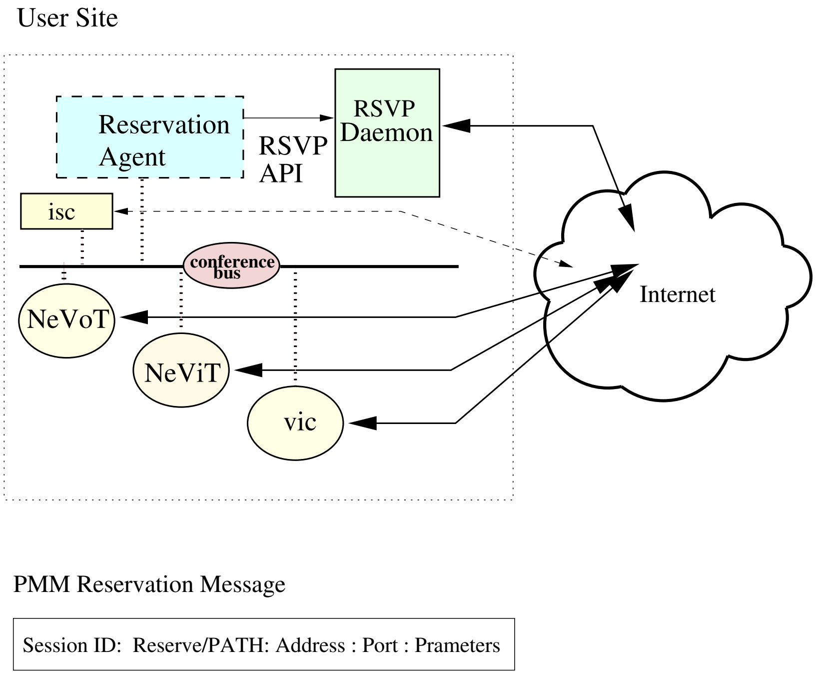


Figure 2: Example multimedia conferencing control architecture

In [27], a communication protocol called Pattern Matching Multicast (PMM) is introduced that can be used to start and terminate media agents. PMM can exchange session parameters such as a unique session identifier and the email addresses of the participants and change parameters of media agents during a session, such as bandwidth and frame rate of a video source or the compression algorithm. The messages used in this protocol are sent as ASCII text and are formatted to be directly interpretable by a standard Tcl interpreter [25].

An agent sends a PMM message to indicate a change in state or to initiate an action in another agent. The name “PMM” derives from the property that messages are not addressed to a particular agent, but rather to a group of agents that have expressed interest in particular commands or events. Just as in the IP host group model [8], senders do not need to be aware which, if any, agent is interested in processing the PMM message. A single message may trigger actions in zero or more agents. There are a number of possible implementations for such a replication mechanism, including a centralized message replicator agent or host-local multicast. The only assumption is that messages are distributed reliably. We have chosen the configuration depicted in Fig. 2, in which messages are exchanged using host-local IP multicast. That is, agents send packets to a well-known multicast group with the time-to-live (ttl) value set to zero, which indicates that data should not be sent outside the host. In another approach, a replicator process listens to a well known TCP port for messages and sends them to all media agents and controllers that have expressed interest in that particular type of message. This latter method has the advantage that only the processes that have expressed interest in a certain message type are woken up instead of all processes subscribed to the multicast address. However, this adds another process, which also needs to keep track of which client processes are still alive.

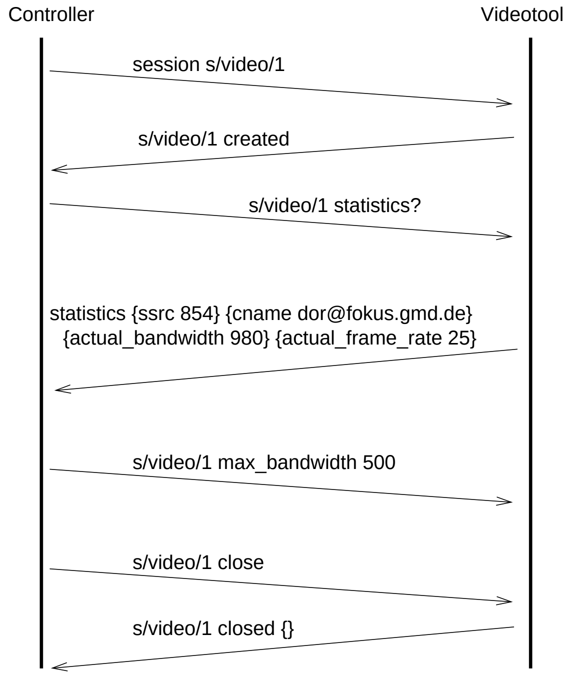


Figure 3: Example for using the conferencing protocol

Fig. 3 shows a simple example for such an interaction between a central conference control agent and a video agent. Each message contains a hierarchical session identification and the message body. The identifier *s/video/1* used in Fig. 3 denotes the conference s, the media type *video* and the instance of the media, *1*. The first message "session" creates a blank media session and starts an appropriate media application, in this case a video tool. The application responds with a "created" message. With the "statistics" message the application is asked to respond with some of the measured values such as the data rate of outgoing or incoming video streams or the loss rate. Receiving a message with the parameter "max_bandwidth" set to 500, the video agent sets its transmission rate limit to 500 kb/s. Finally, the "close" message concludes the media session. In PMM, media agents announce when members join or leave a media session, change their RTCP SDES information or start transmitting data.

The controller application does not need to know if a video agent is handling one or several video sessions. If the session creation message does not elicit a response within a time-out interval, the message sender starts the application and re-issues the request.

Since all parameters of a media agent are controllable through PMM messages, media agents do not need any user interface. All user interaction is handled through a single interface in the session controller. The idea of remote-controlled applications is similar to Microsoft ActiveX controls, except that the PMM mechanism is text-based and the set of receivers is unknown to the sender. Tcl also has a "send" mechanism that allows to execute Tcl commands in Tcl applications using the same X display. However, it is limited to Tcl applications and is, again, restricted to addressing one particular instance of a named application.

Currently, we do not have a mechanism that allows a controller to discover the parameters that are setable for a particular media session. However, this facility could be added readily as part of the "created" response.

4 The Internet Multimedia Terminal (MINT)

In MBone conferences, much of the participant information and control is duplicated across several media agents. Rather than a participant-centric view, i.e., which user is using which media, MBone tools encourage a media-centric view, with each media agent displaying a separate list of participants, often using different ways of identification. It is also difficult to integrate media agents into new, domain-specific applications with their own user interface.

Based on these observations we designed MINT as consisting of several media agents, each having a minimal graphical interface or even none at all. All agents can either work independently or interact with each other using PMM. In addition, a control entity was added to manage and control the different agents by sending PMM messages. This architecture gives the user the impression of using a monolithic tool.

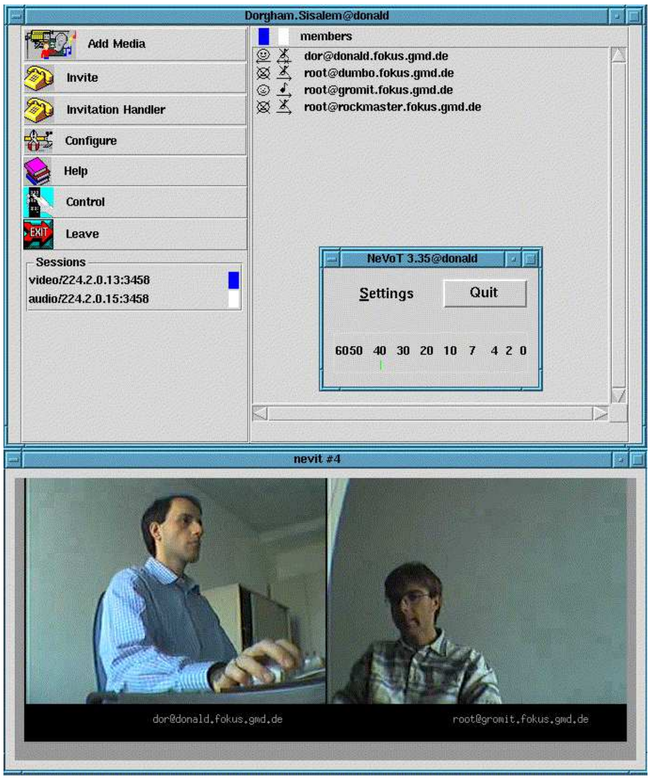


Figure 4: A snapshot of the MINT conferencing tool

Fig. 4 shows a snapshot of some of the of the features of MINT. A central controller called ISC displays the session members and the joined sessions. It provides the user with a graphical interface to start and control the different media agents. The figure shows also the audio and video panels. We will discuss each component in the following sections.

4.1 The Integrated Session Controller (ISC)

The integrated session controller (ISC) is the central control entity in MINT. It provides the user with the necessary interface for initiating and terminating a session. It provides the controls that govern rendering or ignoring the media data received from a participant. It also has control panels for choosing the appropriate parameters for the different media agents. Each user interaction with ISC results in the sending of a PMM message describing the desired action. ISC also displays the names of all session members, the sessions they are listening to and if they are sending or receiving any data. Any media agent that uses the PMM conference bus can be controlled by ISC.

Integrating the configuration panels of the different tools into ISC gives the user a better overview and thereby a better control of the different agents used. Also, as the different procedures of creating, controlling and terminating the different agents integrated in MINT are rather similar, the learning time needed for starting multimedia sessions with MINT is shorter than that needed to learn all the different tools used currently in the Internet multimedia conferences.

4.2 The Network Video Tool (NEVIT)

The first MBone video tools were the Xerox PARC Network Video tool (nv) [11] and the INRIA video Conferencing System (ivs) [38]. While both of these tools were intended for supporting low bit-rate multicast over the Internet, they choose different compression algorithms and video representations. As extending any of the two tools to support the other's compression style was a non-trivial task, both tools were non-interoperable. More recent tools such as VIC [22] support both hardware and software compression engines and offer better performance.

VIC supports a limited form of interaction with the VAT [1] audio tool: VAT can signal speaker activity through a mechanism similar to the conference bus. The speaker activity indication then selects which site is displayed in full size. There has also been some work concerning synchronizing audio and video based on VIC and VAT [20].

We built a new video media agent, NEVIT, that lets us explore the user interaction features of MINT. NEVIT has no graphical user interface at all. Instead, all functions of the tool, including establishing and terminating sessions, configuring the encoding used and setting the transmission parameters are controlled by ISC using the PMM messages sent on the conference bus. Through the messages NEVIT can either communicate with the conference controller or with any other media agent listening to the local conference bus and capable of interpreting the messages.

While our video tool supports different compression algorithms, multicasting and the ability of handling a variable number of video streams, our main goal was to produce a flexible tool that can easily be extended to achieve intermedia synchronization, automatic quality of service control and interaction with other media agents without necessarily being dependent on those agents. The tool can be roughly divided into three parts: routines that process messages arriving on the local conference control bus, routines dealing with network protocols including RTP and video compression/decompression routines.

4.2.1 PMM Implementation Issues

In implementing the conference control bus, some care is necessary to avoid locking out commands from the bus during high-rate video processing. This lock-out is possible due to the use of event-based programs in Unix and X11. More precisely, Unix applications using X11 or processing data from several sources typically use the `select()` event multiplexer as their main loop. When one or more sockets have data waiting to be read, `select()` returns a bit mask with the indices of sockets with data waiting. If the event handler always checks first for the socket on which video data arrives and the CPU can barely keep up with the compression or decoding of the data, the conference control bus socket may never be read as there is always video data waiting. Thus, the conference controller looses control over the application. Unfortunately, with most event handling packages such as the Tcl or Xlib routines, the programmer cannot predict the order of event processing, so that experimentation is required to ensure that the socket handling the conference control bus is always checked first for messages. Alternatively, a threads-based approach can be used, but requires greater care in managing concurrent access to data structures.

4.2.2 Network Interface

NEVIT is based on the real time transport protocol. It uses as a transport protocol unicast or multicast UDP over IP. As an additional feature, NEVIT supports native ATM by sending data directly using AAL5/ATM. This is supported using the application programming interface provided for the FORE ATM-adapter cards based on SPANS. Updating this interface to UNI 4.0 will be one of our future tasks.

4.2.3 Video Handling

Currently, NEVIT only supports the SunVideo card for capturing and compressing video images. The card supports JPEG, MPEG, CellB and YUV video [37] in hardware. NEVIT on the other hand, provides the appropriate algorithms for decompressing and displaying JPEG, MPEG and YUV video images.

The JPEG decoder was ported from the VIC video tool. It is based upon the JPEG decompression code provided by the Independent JPEG Group and enhanced with conditional replenishment. With conditional replenishment, only those parts that have changed in consecutive frames are actually decoded, resulting in a faster decoder and lower processing overhead. On a Sun SPARC 20/712, we manage to receive, decompress and display JPEG frames at the full rate of 30 frames/s. The user-level handling consumes around 70% of the available processor capacity. The network data rate is about 1.3 Mb/s.

YUV denotes the uncompressed video signal consisting of luminance and subsampled chrominance values (4:2:2), requiring four bytes for two pixels. While the rendering of YUV data is simple, a YUV-coded video with 30 frames of 240 by 320 pixels per second generates about 30 Mb/s, making it suitable mainly as a test load for ATM networks. (Compressing this video stream into JPEG frames with a quality factor of 50 yields a data rate of about 1-1.5 Mb/s.)

In addition to that, NEVIT is capable of sending and receiving MPEG-1 video streams using the hardware codec on the SunVideo card for compression and the Berkeley MPEG library to handle the decompression.

4.3 The Network Voice Terminal (NEVOT)

NEVOT [26] is an audio tool that allows the user to join different audio sessions simultaneously. It has only a minimal control interface and is mainly configured using ISC. It supports audio qualities ranging from communication-quality at about 4 kb/s to high fidelity, CD-quality audio at more than 1.5 Mb/s (16 bit stereo at 44.1 kHz or 48 kHz sampling rates). It currently operates over UDP/IP, but can be easily modified to use ATM directly, should that be desirable. The transmission quality of NEVOT can be changed by switching audio encodings during a transmission, with different participants being able to use different encodings at the same time. Just as with NEVIT, NEVOT is based on the RTP/RTCP protocol. Also, it supports the vat protocol which ensures its interoperability with the VAT audio tool from the Lawrence Berkeley Laboratory (LBL).

4.4 The Session Floor Controller (IFLOOR)

In conferences with long latencies and low-frame-rate video, participants lack the visual and auditory cues to negotiate who gets to talk. Experience has shown that in these sessions, turn-taking becomes difficult, with multiple participants jumping in at the same time after a speaker finishes. Participants also have no way to subtly let the current speaker know that they want to speak up, without rudely interrupting. Thus, there tends to be less back-and-forth and more long monologues. In addition, on bandwidth-limited links, multiple simultaneous speakers may well lead to packet loss, rendering them all incomprehensible. For these reasons, floor control is needed even for small discussion groups. It is particularly important for large groups often found in teleteaching applications, as one cannot afford to transmit video for each participant, and thus both the instructor and members of the audience lack any visual cues as to who would like to speak.

We have enhanced MINT with a floor controller that coordinates who is allowed to speak. The controller is designed to work in a decentralized fashion so it could easily scale to support sessions of hundred or more participants. Each participant runs a copy of the floor controller and has the ability to push buttons to "raise or lower her hand". Floor controllers send messages via multicast to all other floor controllers. User interaction triggers the local floor controller to send a time-stamped request to speak or a request to cancel an earlier speaking request. The message also includes the identity of the requesting host and the session identity. A reliable multicast protocol should be used to mitigate the effect of packet losses. Currently, we still rely on a simple multiple-transmission scheme but are working on adding a more robust approach.

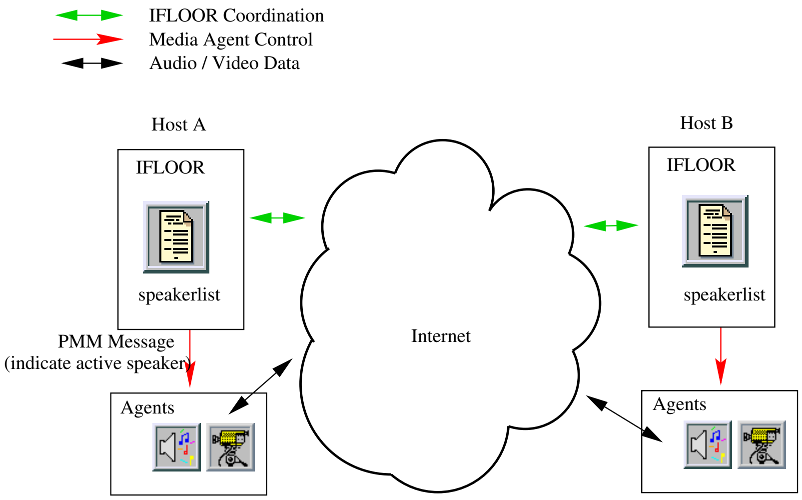


Figure 5: Session floor control architecture

We can distinguish between two modes in which the controller may be used:

Conference-style sessions: In a conference-style session, all conference participants have the same priority. Floor controllers maintain a time-ordered request list. We assume that time is synchronized across session participants to within a few seconds, as is easily accomplished using NTP [24]. Thus, each participant can decide independently that it is her turn to speak.

Moderated sessions: In a moderated session, there is a central moderator that assigns the right to speak to all the requesting members in a centralized way. A teleteaching scenario might use such a configuration with the students requesting the right to ask questions and the teacher deciding whom to take questions from. The request to speak reaches the moderator site via multicast, or is sent to it via TCP. The latter may be preferable in this case, as it avoids lost requests. The moderator can select any of those waiting in line to speak by sending a message indicating his choice. Similarly, the floor is withdrawn with another message. A moderator might be selected through the conference control protocol, e.g., a session directory or conference invitation protocol (SIP) [15].

As Figure 5 shows, the floor controllers involved in a session negotiate which member receives the right to speak. Based on the state indicated by either the moderator or the locally maintained list, the floor controller application sends PMM requests to the local audio and video applications to enable and disable sending and receiving. More precisely, the floor controller disables reception from all members except the speaker and disables sending unless its local list or the moderator indicate that it has the right to speak. To maintain a more interactive mode, we could imagine the case in which the the floor controller does not simply disable the reception of audio data from other receivers but reduces the volume with which their data is being played out. In this manner, a session participant that ignores the advice of its local floor controller intentionally or because of a malfunction cannot disrupt the whole conference, except by injecting excessive traffic into bandwidth-limited links. Note that the media agents are not aware that a floor controller instead of ISC governs sending and receiving.

4.5 Session Initiation Agent

Currently, to invoke a multimedia session with other members, the session initiator would need to contact all the members either by calling them on the phone, sending them email or posting the session information using a global session directory such as the `sdr` multicast directory session tool commonly used in the MBone. The first two methods are cumbersome particularly for large groups, the third requires that listeners know in advance about the existence of the session. Multicast announcements also have scaling problems, as almost all announcements end up being sent to the world at large. Currently, sdr typically advertises on the order of twenty upcoming and on-going sessions; with commercial use, the number could well be larger than the number of TV and radio stations, measured in the thousands.

The Session Initiation Protocol (SIP) [16, 29] being standardized within the IETF describes a method for inviting users to join a session. It also conveys the necessary information to the end system to configure the local media agents. Unlike some of the currently available conferencing applications that require knowledge of the current login address of a participant, the SIP invitation agent uses the email address of the user to be invited to locate his whereabouts and sends them an invitation to join the session. Figure 6 sketches the method used for this location service. Given the email address of the user to be invited, the domain name service (DNS) provides the address of the mail server for the invitee's domain. To start a session, the conference initiator's SIP agent sends a SIP-message to the SIP daemon residing on that mail server. That daemon only needs to locate the invitee within its local domain. There are a number of possible approaches to locating users: A data base updated at login time or periodically can track which host a particular user is currently logged in at. Alternatively, the daemon can multicast a search request on the local network or might use an existing protocol such as the finger protocol [40]. (We have experimented with a recursive version of that protocol, making use of the fact that the finger protocol returns the host where the user last logged in from and following that lead.) Many other methods are conceivable, depending on system capabilities.

Having located the host where the invitee is currently logged in, the daemon on the mail server forwards the invitation to that address. The invited user can then join the session, reject the invitation or just forward the call to some other site.

The invitation messages contain a description of the session using, for example, the Session Description Protocol (SDP) [14]. This protocol is used to describe the media types involved in the session, which coding styles to be used, some QoS parameters and additional information such as when to automatically invoke the session in the future. Having accepted an invitation, the SIP agent contacts ISC and invokes the appropriate tools to join the session based upon the session description messages.

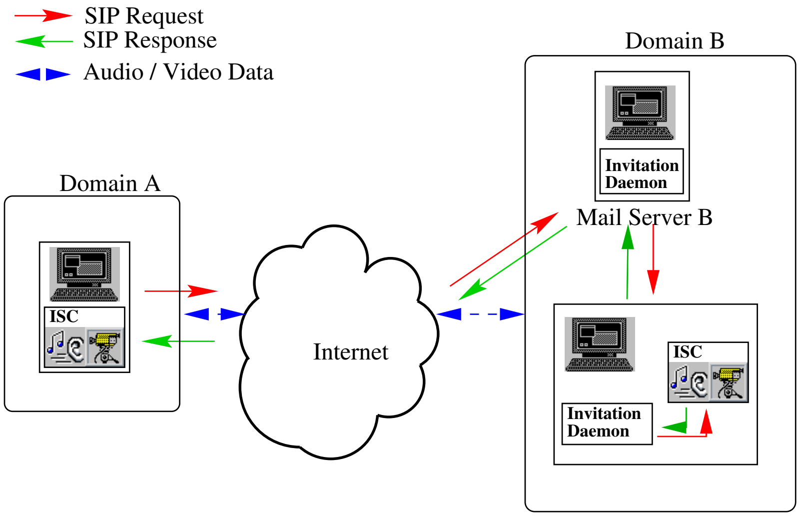


Figure 6: Invitation and initiation of multimedia sessions using SIP

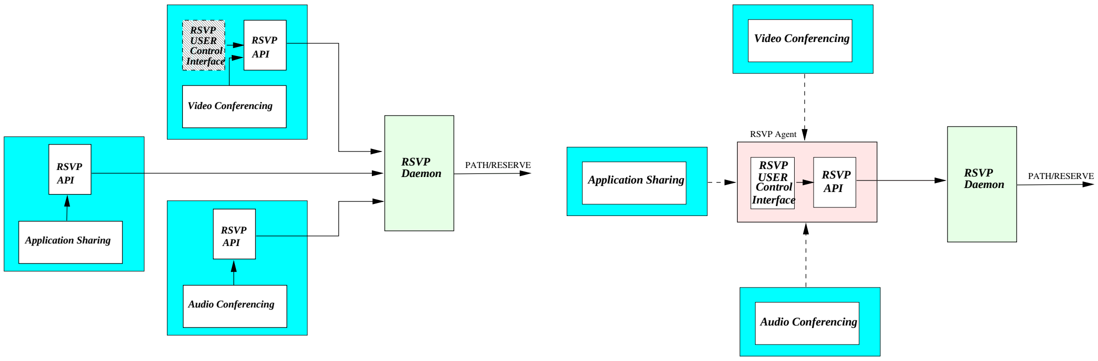
5 Quality of Service Control in MINT

Multimedia conferencing applications have two characteristics in terms of their quality-of-service requirements. On the one hand, audio and video transmissions require a minimal guaranteed bandwidth and, for interactive sessions, an upper bound on the end-to-end delay. On the other hand, video media agents, in particular, can adapt to a wide range of available bandwidths to provide increased perceptual quality. This section discusses some of the QoS control mechanisms added to MINT that provide the user with a better QoS than the best effort QoS level currently supported in the Internet.

5.1 Resource Reservation using RSVP

Bandwidth reservation is essential for multimedia conferencing to guarantee a minimal level of uninterrupted service throughout the session. The IETF has recently standardized RSVP [39, 4] as the protocol for setting up reserved sessions. When using RSVP as a QoS signaling protocol, participating end systems establish a closed control loop. The senders inform the network and receivers about their traffic characteristics by sending PATH packets. The actual reservation is triggered by the receivers who send their reservation requests back towards the sender in RESV messages, based on the traffic profiles announced.

The current approach for using RSVP is to have the application invoke RSVP functionality via an API, as in Fig. 7(a). This API interacts with an RSVP daemon that sends and receives PATH and RESV messages. Additionally, to allow for flexible control of RSVP, the user interface of the application needs to be enhanced as well. While this might work fine for applications available in source code and for users familiar with the applications, it is rather difficult for ordinary users and impossible for applications only available in binary format. In this case, a user wishing to benefit from RSVP would need to wait until a next release of the application is available that supports RSVP.



(a) Current approach for integrating RSVP

(b) Using RSVP with the reservation agent

Figure 7: Usage of the reservation agent

To avoid this restriction, we added a reservation agent to MINT that allows a user to specify the resources it wishes to reserve and which session it wishes to make the reservation for. This is an independent application that is controlled through ISC and offers the user a graphical interface that allows him to specify different RSVP parameters such as filtering types, traffic characteristics, session address and resources to be reserved. Applications that are integrated in MINT can directly communicate their session and QoS parameters to the reservation agent using PMM. Otherwise, for applications wishing to use RSVP but are not integrated in MINT, the user can specify those parameters using the interface of the agent, see Fig. 7(b).

The data flow of reservation requests is illustrated in Figure 8. The session controller sends a “reserve” PMM message containing flow specifications on the local conference bus, which is picked up by the reservation agent. Through a library interface, the reservation agent communicates with the local RSVP daemon, which then forwards RSVP messages into the network.

Applications can announce over the local PMM conferencing bus their traffic description to be used in the PATH messages. The user can, additionally, manually change these

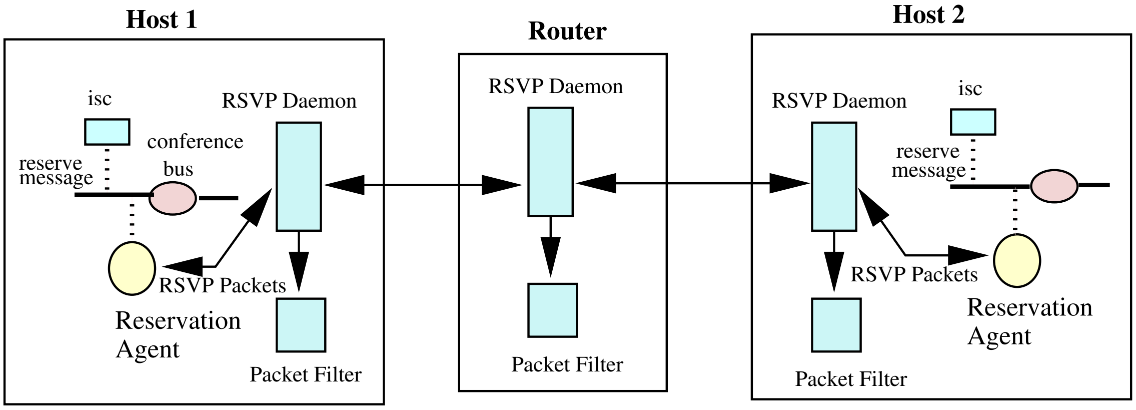


Figure 8: Reservation scheme using RSVP with MINT

characteristics if required. At the receiver sites reservations can be made based on those characteristics.

5.2 Adaptive Applications

For a number of years, reservation will not be supported in large parts of the Internet, in particular since bandwidth reservation requires new charging and settlement mechanisms. Additionally, as it is usually rather difficult to anticipate the exact characteristics of a certain stream in advance one would tend to over-reserve resources to guarantee the requested QoS level. This would, however, lead to under utilizing the network and would get costly if one has to pay for the reserved but unused resources.

With no reservation available, the conference participants or at least the conference organizers have to manually set the appropriate bandwidths for the different media streams, currently with little guidance from the media agents. Instead of guessing at the appropriate bandwidth setting and then suffering or causing unnecessary packet loss, applications should adapt themselves to the available network bandwidth. Also, even if resource reservation is available, it may be desirable to attain the best possible quality of service given current network load and fair sharing of bottleneck bandwidth.

Adaptation schemes are particularly attractive for video streams. Video usually consumes more bandwidth than audio and is of less importance, especially for the cases of teleteaching and personal communication for which MINT was initially designed for. Video is also more flexible in its bandwidth needs and thus lends it self more readily to adaptation. Our video media agent (NEVIT) is enhanced with a bandwidth adaptation algorithm that tunes the video frame rate to achieve different transmission data rates. Adapting bandwidth to current network conditions requires the exchange of state information between the source and the network nodes or the receivers. The first approach is used for the ATM available bit rate service [33]. The approach we are using is based on the RTP control protocol which, as pointed out earlier, provides periodic loss feedback from the receivers to all members. Note that the reporting intervals for each receiver range from five seconds to a minute or more. Thus, load spikes can lead to sustained packet losses or unfairness in bandwidth usage. On the other hand, the bandwidth of audio and video sources should only be adjusted over longer time periods to avoid perceptually annoying rapid quality changes, e.g., rapidly changing frame rates.

Using the loss information sent within the RTCP packets, a sender can estimate the congestion state in the network. Whenever the loss rate reported in the RTCP packets is above a predefined threshold, the source reduces its rate by a multiplicative reduction factor. If the loss rate drops below some predefined threshold, the source increases its rate additively. A detailed description for such an approach and some results for different network topologies can be found in [5]. We have also been considering other approaches that are less oscillatory [34] and others that are more friendly towards TCP connections [35].

Currently, the adaptation schemes we are working on are only implemented in NEVIT. To use these schemes with video agents other than NEVIT without having to include the schemes in the other applications, we are working on an adaptation agent that receives the RTCP messages, takes the adaptation decisions and informs the video agents about the appropriate transmission rate to send with using the PMM interface.

5.3 Hierarchical Data Transmission

With the application control approach a sender determines the appropriate transmission rate based upon the congestion information arriving from the network or the receivers. As IP-based networks such as the Internet traverse paths of widely ranging bandwidths and load factors, such an adaptation approach would, however, perform poorly. In such a heterogeneous environment the conflicting bandwidth requirements cannot be satisfied with one transmission rate. Therefore, the rate is usually adapted to the worst receiver requirements, thereby reducing the quality of the data received at all receiving sites.

To avoid these problems, various proposals have been made for hierarchical data distribution, for example [17, 23]. Those proposals are based on partitioning a data stream into a base layer, comprising the information needed to achieve the lowest quality representation and a number of enhancement layers. The different layers are then sent to different multicast sessions. Based on their capacities, receivers can determine how many sessions to join and thereby adapting the quality of the received data to their own requirements.

As video streams are usually the most demanding type of data in terms of bandwidth a wide range of hierarchical video encodings have already been proposed [32]. In NEVIT, we have implemented a simple layering scheme based on partitioning the frame rate among the available layers. This is the simplest form of data layering that entails only little added complexity to the end systems. Joining or leaving a session can either be controlled manually or using an approach similar to that described in [17] based on reserving enough resources for each layer before actually joining it.

6 Summary and Future Work

In the work presented here, we described a multimedia tool (MINT) based on loosely integrated media agents with a central control entity. The different agents communicate with each other and with the control agent using a simple communication protocol called PMM. The tool supports audio and video conferencing and offers the user some other conferencing control agents such as reservation, invitation and control agents as well as QoS monitoring capabilities.

To improve the QoS of the conferencing sessions MINT supports the RSVP protocol for making resource reservations. The video part of the tool (NEVIT) can adjust its transmission rate in accordance with the network congestion state based on the feedback messages sent with the RTCP protocol. Finally, to accommodate heterogeneous receivers, video streams can be sent in separate layers, allowing the receivers to join the number of layers suitable for their capacities.

Among our future tasks we are planning a voting agent, with which a user can distribute a proposal among the session members and collect their votes on the proposal. This agent must naturally handle data transfer reliability and security issues. All members of a session must receive the entire proposal and all the votes must be correctly collected. Also, proposals as well as votes should not be falsified resulting thereby in taking the wrong decisions.

To extend the supported video encodings of MINT we have integrated the VIC video tool into MINT. By stripping VIC of its user interface and replacing it with a PMM message handling interface, we can use VIC as an integral part of MINT.

We are also working on integrating MINT with a video conference recording tool called the MBone video conference recording on demand (MVCRoD) [18]. This would give the user the possibility of recording the audio and video streams generated during a conferencing session and playing them out later.

A major issue that we have not been able to handle yet, is the support of application sharing. For the distribution of slides and joint editing we are still using the WB [10] white board from LBNL. As the application is only available in binary format we are not able of integrating it into MINT. However, the task of supporting application sharing will be one our major future goals.

MINT is currently being used for testing various aspects of multimedia conferencing such as reliability and usability in various scenarios ranging from project meetings to teleteaching.

7 Acknowledgments

We would like to thank all those who participated in implementing the different parts of MINT: Stefan Hoffman for implementing the invitation agent, Frank Emanuel for his work on the integration of VIC, Ilona Schubert for the implementation of the floor controller and Krzysztof Samp who implemented the reservation agent. We also appreciate the work and comments of Henning Sanneck and all participants in the USMInT project. Finally, we would like to acknowledge the anonymous reviewers, who provided many comments that were valuable in revising an earlier version of the paper.

References

[1] S. Baker. Multicasting for sound and video. *Unix Review*, 12(2):23–29, Feb. 1994.

[2] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 58–67, London, England, Aug. 1994. ACM.

[3] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. Technical Report RFC 1633, Internet Engineering Task Force, June 1994.

[4] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation protocol (RSVP) – version 1 functional specification. Technical Report RFC 2205, Internet Engineering Task Force, Sept. 1997.

[5] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS control of multimedia applications based on RTP. *Computer Communications*, 19(1):49–58, Jan. 1996.

[6] S. Casner and S. Deering. First IETF Internet audiocast. *ACM Computer Communication Review*, 22(3):92–97, July 1992.

[7] M. Cohen and N. Koizumi. Exocentric control of audio imaging in binaural telecommunication. *IEICE Transactions on Fundamentals*, E75-A(2):164–170, Feb. 1992.

[8] S. E. Deering. *Multicast routing in a datagram internetwork*. PhD thesis, Stanford University, Palo Alto, California, Dec. 1991.

[9] H. Eriksson. MBone – the multicast backbone. In *Proceedings of the International Networking Conference (INET)*, pages CCC–1 – CCC–5, San Francisco, California, Aug. 1993. Internet Society.

[10] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. Reliable multicast framework for light-weight sessions and application level framing. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages –, Cambridge, Massachusetts, Sept. 1995.

[11] R. Frederick. Experiences with real-time software video compression. In *Sixth International Workshop on Packet Video*, Portland, Oregon, Sept. 1994.

[12] M. Handley. SAP: Session announcement protocol. Internet Draft, Internet Engineering Task Force, Nov. 1996. Work in progress.

[13] M. Handley, J. Crowcroft, C. Bormann, and J. Ott. The internet multimedia conferencing architecture. Internet Draft, Internet Engineering Task Force, July 1997. Work in progress.

[14] M. Handley and V. Jacobson. SDP: Session description protocol. Internet Draft, Internet Engineering Task Force, Mar. 1997. Work in progress.

[15] M. Handley, H. Schulzrinne, and E. Schooler. SIP: Session initiation protocol. Internet Draft, Internet Engineering Task Force, July 1997. Work in progress.

[16] M. Handley, H. Schulzrinne, and E. Schooler. SIP: Session initiation protocol. Internet Draft, Internet Engineering Task Force, Mar. 1997. Work in progress.

[17] D. Hoffman and M. Speer. Hierarchical video distribution over internet-style networks. In *ICIP'96*, Lausanne, Switzerland, Sept. 1996.

[18] W. Holfelder. Interactive remote recording and playback of multicast videoconferences. In *4th. International Workshop on Interactive Distributed Multimedia Systems and Telecomminication Services (IDMS '97)*, Darmstadt, Germany, Sept. 1997.

[19] N. Kanemaki, F. Kishino, and K. Manabe. A multi-media teleconference terminal controlling quality of flow in packet transmission. In W. A. Pearlman, editor, *Visual Communications and Image Processing IV*, volume 1199, pages 259–266, Philadelphia, Pennsylvania, Nov. 1989. Society of Photo-Optical Instrumentation Engineers.

[20] I. Kouvelas, V. Hardman, and A. Watson. Lip synchronization for use over the internet: Analysis and implementation. In *GLOBECOM'96*, London, UK, Nov. 1996.

[21] S. Masaki, T. Arikawa, H. Ichihara, M. Tanbara, and K. Shimamura. A promising groupware system for broadband ISDN: PMTC. *ACM Computer Communication Review*, 22(3):55–56, Mar. 1992.

[22] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proc. of ACM Multimedia '95*, Nov. 1995.

[23] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *SIGCOMM Symposium on Communications Architectures and Protocols*, Palo Alto, California, Aug. 1996.

[24] D. L. Mills. Network time protocol (version 3) specification, implementation. Technical Report RFC 1305, Internet Engineering Task Force, Mar. 1992.

[25] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, Reading, Massachusetts, 1994.

[26] H. Schulzrinne. Voice communication across the Internet: A network voice terminal. Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.

[27] H. Schulzrinne. Dynamic configuration of conferencing applications using pattern-matching multicast. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science (LNCS), pages 231–242, Durham, New Hampshire, Apr. 1995. Springer.

[28] H. Schulzrinne. Simple conference invitation protocol. Internet Draft, Internet Engineering Task Force, Feb. 1996. Work in progress.

[29] H. Schulzrinne. A comprehensive multimedia control architecture for the Internet. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St. Louis, Missouri, May 1997.

[30] H. Schulzrinne. Re-engineering the telephone system. In *Proc. of IEEE Singapore International Conference on Networks (SICON)*, Singapore, Apr. 1997.

[31] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. Technical Report RFC 1889, Internet Engineering Task Force, Jan. 1996.

[32] N. Shacham. Multipoint communication by hierarchically encoded data. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, volume 3, pages 2107–2114 (9A.4), Florence, Italy, May 1992. IEEE.

[33] S. S. Shirish. ATM Forum traffic management specification version 4.0. Technical Report 94-0013R6, ATM Forum, June 1995.

[34] D. Sisalem. End-to-end quality of service control using adaptive applications. In *IFIP Fifth International Workshop on Quality of Service (IWQOS '97)*, New York, May 1997.

[35] D. Sisalem, H. Schulzrinne, and F. Emanuel. The direct adjustment algorithm: A tcp-friendly adaptation scheme. Technical report, GMD-FOKUS, Aug. 1997. Available from http://www.fokus.gmd.de/usr/sisalem.

[36] D. Sisalem, H. Schulzrinne, and C. Sieckmeyer. The network video terminal. In *HPDC Focus Workshop on Multimedia and Collaborative Environments (Fifth IEEE International Symposium on High Performance Distributed Computing)*, Syracuse, New York, Aug. 1996. IEEE Computer Society.

[37] Sun Microsystems. *SunVideo User's Guide*. Sun Microsystems, Mountain View, California, Aug. 1994.

[38] T. Turletti. H.261 software codec for videoconferencing over the Internet. Rapports de Recherche 1834, Institut National de Recherche en Informatique et en Automatique (INRIA), Sophia-Antipolis, France, Jan. 1993.

[39] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: a new resource reservation protocol. In *Proceedings of the International Networking Conference (INET)*, pages BCB–1, San Francisco, California, Aug. 1993. Internet Society.

[40] D. Zimmerman. The finger user information protocol. Technical Report RFC 1288, Internet Engineering Task Force, Dec. 1991.