A comprehensive multimedia control architecture for the Internet*

Henning Schulzrinne
schulzrinne@cs.columbia.edu
+1 212 939 7042
Dept. of Computer Science
Columbia University
New York, NY 10027

Abstract

The Internet and intranets have been used to deliver continuous media, both stored and live, for a number of years. Most of the attention has focused on providing guaranteed quality of service (RSVP) and end-to-end data transport (RTP), with every application using its own control protocol. In this paper, we describe a control architecture that offers most standard advanced telephony features and integrates stored and conference multimedia. The protocol re-uses much of the "infrastructure" of HTTP, including its security and proxy mechanisms. The architecture is instantiated by two related, but independent protocols: the Session Initiation Protocol (SIP) for inviting participants to a multimedia session and the Real-Time Stream Protocol (RTSP) to control playback and recording for stored continuous media.

1 Introduction

In the last few years, the Internet has become a viable infrastructure to support multimedia services, including one-way retrieval of stored multimedia content on demand, live, broadcast-like live events and real-time interactive services such as telephony and conferencing.

Stored, live and conferencing multimedia services require a number of new Internet services for both data transport and control. Protocols for transporting real-time data [1] and for reserving resources to guarantee quality of service [2] have been developed, standardized and are beginning to be widely used in products. However, protocols to initiate and control multimedia sessions are less well developed. In this paper, we present two independent, but interacting protocols that initiate and control stored, live and interactive multimedia sessions in the Internet. They leverage the evolving infrastructure of the world-wide web and also provide it with a major missing ingredient, namely interoperable continuous media services. The protocols support the following scenarios:

Phone call: Two-party and multi-party multimedia calls with standard telephone services such as call forwarding, automatic call distribution, third-party signaling and answering services.

Invitation to a multi-party conference: It is possible to invite users to sessions announced through a multicast session directory [3] or a web page.

Near video-on-demand: Popular movies are shown staggered in time across a set of multicast groups and may be controlled by a teacher, for example.

Video-on-demand: Movies are requested by individual viewers and can be controlled in VCR-like fashion. Movies can have several audio and video tracks, to be played alternatively or in parallel, possibly located at different places.

Virtual presentations: A client can assemble synchronized multimedia presentations from a distributed set of servers, without the individual servers having to be aware of that fact.

Distributed digital editing: With content possibly distributed among multiple sites, new content can be created and stored at yet another location.

Combining stored, live and interactive multimedia: A media server can be invited to a conference or phone call and play into this interactive session or record the session, controlled by one or more conference participants. There are many applications for this, such as a group of people watching a movie or training video together.

*This work was supported in part by a grant from the AT&T and Lucent Foundation.

We describe two protocols that support the scenarios described above, namely the Session Initiation Protocol (SIP) [4] in Section 4 to establish and control multimedia conferences and the Real-Time Stream Protocol (RTSP) [5] in Section 5 to control delivery of stored and live streaming multimedia content. In addition, we briefly mention efforts towards an improved description format in Section 6. Section 2 motivates the protocol design. While this set of protocols enables interoperable deployment of a wide variety of Internet multimedia services, Section 8 alludes to some pieces which are still missing.

1.1 Related Work

Efforts to design multimedia applications and protocols for packet-switched networks [6, 7] date back to the early days of the Internet; systems have been developed for various combinations of packet-switched and circuit-switched networks [8, 9]. In particular, the set of tools commonly known as the Mbone conferencing tools [10, 11, 12, 13] has achieved widespread use. Examples of multimedia control include Etherphone [14], Rapport [15] and MMCC [16, 17].

Delivery of stored multimedia content in a streaming mode, that is, playing back the multimedia content as it arrives rather than downloading the whole presentation, has been commercialized successfully by a number of companies, however with proprietary and limited control functionality. In the telecom-oriented version of video-on-demand, a control protocol called DSM-CC [18] has been specified by the DAVIC consortium for transport of MPEG streams over a broad range of ATM and CATV networks. RTSP borrows one of the time concepts from DSM-CC, but, in the tradition of Internet protocols, does not depend on a whole set of supporting protocols. Unlike DSM-CC, RTSP also offers recording and device control and, due to its state machine, is more suited for remote digital editing.

For conferencing and telephony, H.323 [19, 20] offers a basic two-party signaling protocol built on top of the Q.931 ISDN call signaling protocol. However, it is complex, requiring about 300 pages of specifications, not including ASN.1. The complexity is evident in the high latency for call setup, particularly in the wide area, since it needs to establish TCP control connections for H.225.0 and H.245, with application-level handshakes on each. SIP only requires a single round-trip time in the common case where caller and callee simply indicate their capabilities to each other. H.323 also does not support security (authentication and key exchange), or multipoint signaling.

While little of the basic control functionality described in this paper is fundamentally new, the protocols described here for the first time offer an integrated, standardized architecture that ties in with other Internet protocols, in particular, email and the web.

2 Protocol Design

In the past, upper-layer Internet protocols evolved largely independently, with little re-use of syntax and semantics between protocols. (For example, ftp, SMTP, NNTP, POP and IMAP all are text-based protocols exchanging data between clients and servers across TCP connections, yet they allow little reuse of security features, for example.)

It appears that during the initial discussions for each new application-specific Internet protocol, there is a heated debate on the general format of the protocol, namely, "Internet-style" binary, ASN.1, text-based or layering on top of an RPC mechanism like CORBA or DCOM. Here, Internet-style binary refers to C-like structures with elements aligned on word size multiples or type-length-value tuples. In addition to the core Internet protocols, RTP, RSVP and the RADIUS accounting protocol [21] are examples of this approach. This works well as long as protocol requests are flat lists of integers, with few optional parameters and variable-sized structures.

ASN.1 allows the specification of nested data structures with optional elements and a wide variety of basic data types. It can be storage-efficient if the packed encoding rules (PER) are used. The basic encoding rules, which have the advantage of being self-describing in terms of data types, are fairly verbose. Parsing is cumbersome. The only ASN.1-based protocols in widespread Internet use are SNMP and H.323.

Most current Internet application protocols including NNTP, SMTP, ftp, and HTTP are text-based. The parameter-value structure works well where parameters are not structured and values are simple lists, possibly modified by attributes. Binary data is not important for the control protocols discussed here, but can, with loss of efficiency, be carried encoded as base 64. A general parser for headers of that type can be implemented in about 500 lines of C, far less than a general ASN.1 parser. Textual formats are generally less space-efficient than ASN.1 PER or Internet binary formats, but for both protocols discussed here, the number of data bytes exchanged is likely to far exceed those produced by the control protocol. However, space efficiency is still a concern, as it is highly desirable to avoid UDP packet fragmentation. This limits the maximum message size to 1500 bytes.

While the author is not aware of any performance comparisons, it is anticipated that for both SIP and RTSP, the header parsing and space overhead would be a very small. However, the largest advantage is the low cost of entry, since simple client and server implementations can be rapidly built using scripting languages such as Perl or Tcl whose "natural" data type is text. Unlike ASN.1 and Internet binary, headers are self-describing, simplifying debugging and extensions. On the downside, HTTP and SMTP implementations have suffered from a number of security breaches when implementations made unwarranted assumptions about the maximum length of header fields.

In the past, text-based protocols were restricted to US-ASCII or, at best, ISO 8859-1 (for HTTP); SIP and RTSP are not burdened by this legacy and can express any ISO 10646 (Unicode) character in the UTF-8 encoding [22]. (UTF-8 is a variable-length character set encoding that is upward compatible with US-ASCII.)

The final design alternative is to recognize that most control functionality can be modeled as remote-procedure calls. Thus, systems like the OMG's CORBA or Microsoft's DCOM could provide the underlying foundation, removing the need for each new protocol design to specify data representation and transport reliability. Indeed, one could probably replace most of the Internet application-layer protocols such as HTTP, NNTP, SMTP, LDAP and ftp with CORBA implementations. It appears unlikely for this to happen any time soon, because of the relative immaturity of current implementations and their lack of interoperability or widespread cross-platform availability. For reasons that deserve study but are beyond the scope of this paper, RPC protocols have never been widely used for general-purpose applications beyond NFS. The principal additional deterrents in our case were the lack of security support and the high cost of entry.

Based on the discussion above, a text-based approach was chosen for the design of SIP and RTSP. Rather than inventing a new protocol representation from whole cloth, reusing the most successful Internet protocol, HTTP, seemed the more appropriate choice. By using HTTP as a base, the protocols can immediately re-use a number of evolving protocols for electronic commerce [23], authentication [24], content labels and client-side access control [25], protocol extensions [26], state management [27] and content negotiation [28]. Also, servers, proxies and firewalls, all already tuned for high performance, manageability and reliability, can be easily modified to accommodate these new protocols. The commonality between SIP and RTSP also simplifies implementations as many clients and servers can be expected to implement both, given the scenarios described in Section 1.

It has been suggested that one could just extend HTTP/1.1 [29] by adding new headers to existing methods. However, none of the existing methods fit particularly well with the out-of-band control of streaming media. Also, it is likely that web servers and multimedia-on-demand servers will, in many cases, remain distinct, so that burdening a media server with the whole complexity of HTTP incurred by caching and maintaining backward compatibility to earlier versions of HTTP is not warranted.

While RTSP and SIP try to leverage the HTTP infrastructure, they clearly are not object retrieval protocols. This is particularly apparent in terms of caching, where caching responses of RTSP and SIP requests makes no sense, since the actual data is carried "out-of-band", e.g., as an RTP data stream (see Section 5.1).

Another important difference to HTTP is that end points have to maintain state across RTSP requests. Unlike other protocols with an out-of-band control channel like ftp or telnet, RTSP does not tie the RTSP-level session ("connection") to a TCP session. A session is defined only through a globally unique identifier chosen by the server. A client may choose to stay connected to the server for the whole RTSP session or issue each request on a new TCP connection or as a UDP packet, as described below. This is necessary if UDP is to be used without completely rebuilding a TCP-like connection establishment mechanism, but it also greatly simplifies moving stream control between different participants in a conference.

HTTP assumes a reliable byte stream protocol such as TCP as its underlying transport mechanism. For control protocols, the use of TCP simplifies client implementations. (For example, a simple design can use `inetd` to have a process handle exactly one control connection.) TCP also makes it possible to use transport-level security protocols such as SSL.

On the other hand, a flow- and congestion-controlled protocol with strict reliability may not be particularly appropriate for multimedia signaling protocols. TCP timer back-off, for example, may delay retransmission more than necessary, particularly when packet losses are high. Given the query-response nature and low control bandwidth, flow and congestion control are not particularly helpful. Also, for call distribution, user location, group control and awareness, multicasting of control messages is very valuable. Thus, both SIP and RTSP can use either TCP or UDP. For UDP, a request sequence number and timestamp[1] are added to the HTTP-like request and response lines. Unlike TCP, data throughput and flow control are not of concern for these control protocols. While the protocols allow pipelining of requests, i.e., sending a number of requests without waiting for the first one to be acknowledged, this is expected to be needed infrequently. Also, an RTSP or SIP response is needed for each request in any event. Thus, acknowledgments are for one request

[1] Adding a timestamp avoids the need for Karn's algorithm [30] in estimating round-trip times. While almost all requests are idempotent, some like RTSP PLAY are not, so that a per-message sequence number is not sufficient.

only rather than a window of data.

HTTP is asymmetric, with clients issuing requests to servers. In both RTSP and SIP, this is mostly true, however, there are a number of occasions where the server needs to contact the client. For RTSP, the server may want to tell the client about a new media stream, e.g., from an additional camera, that has become available during a live presentation. However, a functional media server application can be written such that only the client issues commands, making it easy to extend existing web servers into continuous media servers. In SIP (with extensions proposed in Section 4), either caller or callee have the ability to terminate or redirect a call.

3 Internet Conferencing and Media-on-Demand Architecture

In contrast to other conferencing architectures, members of Internet conferences are not "connected" at the level of a conference control protocol. A *media session* is defined by the membership in a multicast group or a two-party UDP port/address association. Members are identified by their RTP CNAME [31]. A *session* consists of several media sessions and exists only as a common abstraction in each participant, not in a central registry. A conference is an example of a session. Not all participants in a conference need to be in every media session. Access control is through encryption. This session model is often called the "light-weight session model" [32]; unlike central-registry models, it scales to very large conferences and survives network partitions. Currently, these sessions and their attributes are announced using a well-known multicast address and a simple textual description (SDP) [3], which is also used to arbitrate use of the dynamically assigned multicast addresses. This directory model is well suited for public and private pre-planned group events, but does not support telephone calls or inviting participants to a session, nor does it deal with controlling multimedia streams. The "light-weight" conferences depend on the availability of receiver-oriented multicast, where new receivers do not have to acquire the current list of participants.

4 SIP: Conference Session Initiation and Call Control

Conference control applications use SIP to invite humans and media servers into a multicast conference or establish a two-party phone call. The conference initiation phase has to accomplish three goals:

1. locate the terminal (phone, workstation, mobile phone, answering machine, ...) where the called party can be reached,
2. agree on a set of media and possible encodings for communication,
3. determine if the called party wants to be reached.

A call can fail at any of these stages; one can argue whether agreement or reachability should be determined first. SIP supports all three phases, but it may hand off to another protocol for any of these. In particular, SIP may be used to only choose the terminal type, with H.323 or ISDN signaling handling the call establishment.

To be invited and identified, the invitees have to be named. Since it is the most common form of user addressing in the Internet, SIP chose an email-like identifier of the form "user@domain" or "user@IP_address". The domain name can be either the name of the host that a user is logged in at the time, an email address or the name of a domain-specific translation service. A user at a specific host will be derived through zero or more translations. A single externally visible address may well lead to a different host depending on time of day, media to be used, and any number of other factors. In many cases, the address will be derived through a directory service such as LDAP [33], but it may well be a click on a `mailto` URL.

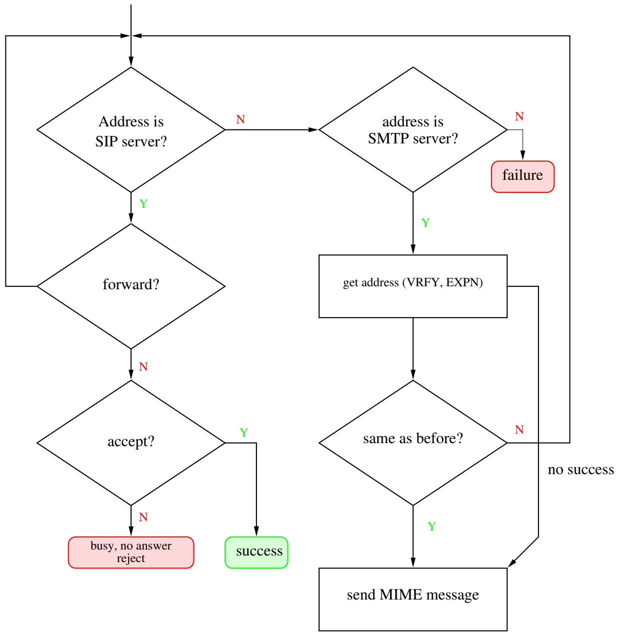

Figure 1: SIP address resolution

The name resolution mechanism is shown in Fig. 1. At every step, the client attempts to resolve the host name via the DNS service (SRV) records [34] first, then checks whether the domain name refers to a physical host and finally checks whether it is a mail exchange host. At each host, the client tries to contact a SIP server; if that fails, it tries to connect to an SMTP server and, if successful, uses SMTP commands to obtain alternate addresses. If all else fails, an email message with the invitation can be sent. If the called party is not at the SIP server named, the SIP server may know where the called party might be located and issues a redirection response, as indicated in Fig. 2 and the client tries again with the new address. Alternatively, the server can act as a proxy and issue an invitation (Fig. 3). The latter case is particularly appropriate for firewalled intranets.

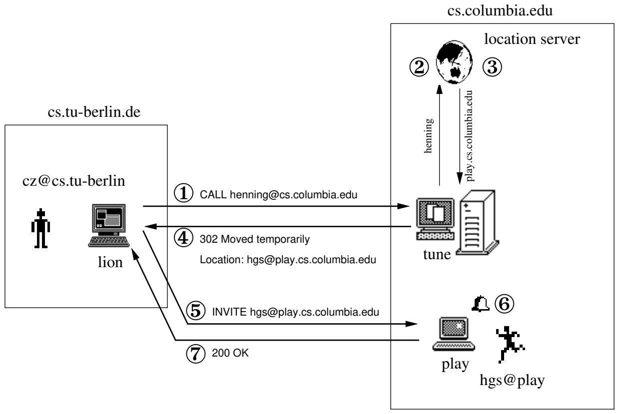

Figure 2: SIP redirection

This naming mechanism, combined with redirection, readily supports “personal mobility”[2]. It has been argued that this functionality should reside with directory services, e.g., X.500, ULS [36] or some other LDAP-accessed directory. While SIP does not preclude doing all user location through a directory service, using SIP has a number of advantages. For example, the answer to the redirection request may well depend on the urgency, time of day or source of the call invitation, not just the name.

The call handling intelligence can be located at either a service provider “in the network” offering a permanent address, at a corporate gateway or at the user’s workstation or some combination of the three. Compared to advanced intelligent networks (AIN) [35], this offers greater flexibility, privacy and user control. In particular, some information such as a person’s schedule or room occupancy should only be available locally.

[2]“Personal mobility is the ability of end users to originate and receive calls and access subscribed telecommunication services on any terminal in any location, and the ability of the network to identify end users as they move. Personal mobility is based on the use of a unique personal identity (i.e., ’personal number’).” [35, p. 44].

Calls are identified by a globally unique conference identifier, consisting of a timestamp and host name, which is created by the conference originator. Every signaling message contains such an identifier, so that proxies maintain state only for a single request, not the whole call or conference. If forwarded through proxies, SIP requests record their route, so that responses can find their way back to the source of the request. This also makes it possible for the called party to send a request to the calling party.

SIP does not address other aspects of conference control, such as floor control, but it can be used to introduce these protocols.

4.1 Choosing Terminals and Locating Callees

Many people have several ways of being reached, including a telephone, email, fax, or a pager, each with widely differing media handling capabilities [37]. Borrowing the concept of HTTP transparent content negotiation [28], a SIP server can return a descriptive list of alternative terminals, their capabilities and addresses.

In a local area, a person may move around from terminal to terminal, e.g., from lab to office to meeting room. Also, a call may be addressed to more than one individual (e.g., a whole department or any member of the sales division). Since SIP can also work over a connectionless transport protocol, it can multicast a “search” for a particular party, to which one or more individuals can respond.

4.2 Negotiating Media Types and Encodings

The SIP INVITE request to join a conference or phone call contains a listing of the media types and associated encodings that the calling party is willing to use. The called party simply responds with a subset of media types and encodings that it is willing to use. Thus, in most cases, negotiation incurs no further delay.

This one-shot negotiation fails to work when a multi-party conference is to be set up, as the session description agreed upon between the conference initiator and the first callee may not be applicable to all participants. Thus, SIP adds an OPTIONS request by which the organizer of a conference call can inquire about a terminal’s capabilities without actually initiating a session.

There may be a need for a more expressive capability language similar to H.245 that can describe asymmetric capabilities (where a terminal can send but not receive a certain encoding), or where only certain combinations of audio and video codings are possible. The latter case may occur if capability sets correspond to different multimedia applications or they may express CPU or screen real-estate

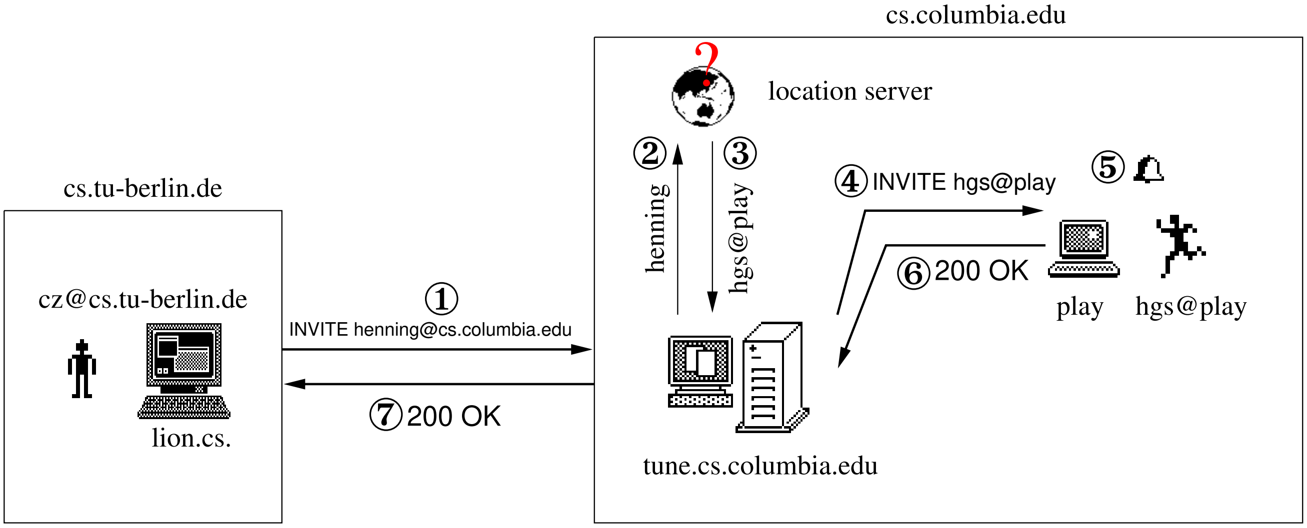

Figure 3: SIP proxying

limitations. With software codecs, the likelihood that elaborate capability descriptions and negotiation protocols [38] are needed, however, appears to be much lower.

4.3 In-Call Signaling

Currently, SIP methods only support call initiation and a limited form of media negotiation, described above. Even so, the standard telephony services of call forwarding, call waiting, caller ID, camp-on, call park and call pickup can be supported.

The services of call termination and call transfer require additional methods. Without SIP support, conference participants simply send RTCP BYE messages and drop their multicast membership when they want to leave the conference. However, BYE messages are unreliable. Thus the author proposes that a call is terminated by having either party issue a BYE request. Call forwarding is a straightforward extension: the party that wants to transfer the call sends a BYE request containing the new destination to the other party. Signaling is greatly simplified compared to standard telephony procedures since the number of simultaneous calls is not limited. Also, a logical call established by SIP may or may not have reserved resources set aside for it at any given time.

5 RTSP: Control of Stored and Live Multimedia

RTSP initiates and controls delivery of stored and live multimedia content to both unicast and multicast destinations. The basic operation of RTSP is illustrated in Fig. 4. The client obtains a description of the multimedia presentation consisting of several media streams, such as the movie description shown in Fig. 6. The description can, for example, be retrieved by HTTP or ftp, be contained within VRML data or a scripting language, sent via email or stored on a CD ROM. The format of the presentation description is outside the scope of RTSP[3]. It is likely that different formats will emerge, with a range of complexity and capabilities. For live sessions, SDP might be used. The browser invokes a helper application based on the content type of the presentation description. Note that the server does not have to be aware of the presentation description, so that synchronized "virtual presentations" can be created where each media type resides on a different server, as long as these servers share a common clock. (While RTSP does not make any assumptions about the protocol used to carry the multimedia data, RTP makes this type of synchronization possible, as it associates media timestamps with an absolute or wall clock time reference.)

Regardless of the description format, each media stream is identified by a new URL method, rtsp://, which can refer to either a single media stream or a presentation consisting

[3] The terms session description and presentation description are used by SIP and RTSP and differ in scope and functionality, as indicated in Section 6.

of several streams. Due to unresolved issues about naming and the interaction with presentation descriptions, presentations where each stream sends to a different network port are currently not supported; also, all media streams in a presentation can only be controlled together, i.e., just pausing the audio is not possible in this arrangement. As in HTTP, the hierarchical naming structure does not necessarily correspond to a hierarchical directory structure.

Unlike HTTP, a client will typically continue to interact with a stream once it has been started, e.g., to pause or fast-forward the stream. Thus, RTSP requires the notion of a session. As explained in Section 2, maintaining a single TCP connection for the duration of delivering a presentation is too constraining. Thus, RTSP defines a session identifier, chosen by the server. The client initiates a session with the SETUP request, which optionally returns an opaque session identifier and the transport parameters actually chosen by the server. If available, the client simply repeats that session identifier for each request that applies to that particular media stream, until the client closes the session with the TEARDOWN request. Instead of this session identifier, a server may also employ dynamic RTSP URLs, so that each session description retrieved has a unique URL. The SETUP request also indicates where the server is to send the data, if not provided in the presentation description. For unicast delivery, the server should still allow specification of the destination port since firewalls may restrict traffic to certain port ranges. For integration into conferences (Section 5.2), the client may also specify a destination unicast or multicast address. This feature must be used with care (and authentication) to prevent remote-controlled denial-of-service attacks.

Load balancing is even more important for media servers than for web servers since the resource commitment is typically larger and lasts longer. Using standard HTTP responses, a server can ask a client to connect to a different server which may be less loaded or topologically closer to the client. RTSP does not specify how this decision is to be made.

The presentation itself can be controlled with PLAY, RECORD and PAUSE. PLAY supports absolute positioning in the logical play time of the media stream. Time can be specified as either SMPTE timestamps, that is, hours, minutes, seconds and frames, or as "normal play time" measured in seconds and microseconds. Ranges of time, with automatic pausing after the range has been played, can be specified, simplifying remote editing and applications like assembly of presentations from stock footage. PLAY requests are queued by the server, rather than preempting the current play as in DSM-CC. Again, this makes remote editing possible. The Scale header field of the PLAY request causes the server to deliver media at other than the normal play rate, e.g., for fast-forwarding. A PAUSE request halts evolution of play time at the first opportunity or at a designated point in time. For example, if the client issues a request to play seconds 10 to 15, then seconds 20 to 30 and finally seconds 10 to 15 again, and then pauses at second 12, only the first segment will be played.

To allow the virtual presentations described earlier, PLAY requests can be scheduled for a particular wall clock time, so that all servers start delivering media content to within their clock synchronization offset, regardless of any transmission, processing or staging delays. This feature also helps with cueing stored presentations into a live conference or broadcast.

In addition, the RTSP requests SET_PARAMETER and GET_PARAMETER may be used to change coding parameters or for device control applications such as remote-controlled motorized video cameras. The protocol does not currently specify the set of parameters to be used.

The client can ask for a description of a media stream with the DESCRIBE command; the server can push a new description to the client using SESSION.

A client may inquire via the OPTIONS request as to which requests the server supports for a particular RTSP URL. In connection with the HTTP protocol extension protocol (PEP) [26], this facilitates future protocol extensions.

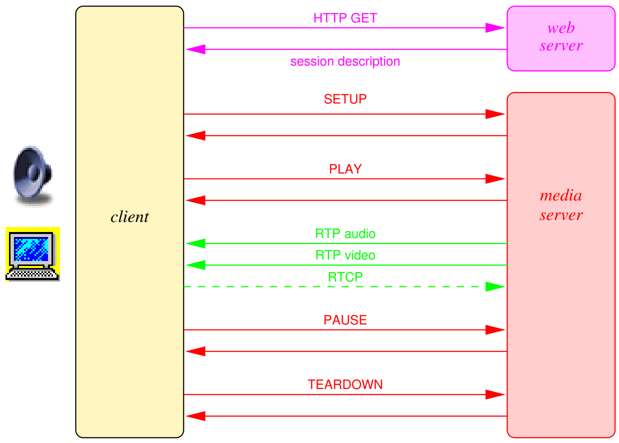

Figure 4: RTSP operation

Authentication, encryption, content labeling and payment are handled by standard HTTP mechanisms. If forced to by ill-designed firewalls, the control stream may be interleaved with the audio or video data.

Since RTSP does not rely on lower-layer transport connections to maintain state, but rather on a server-issued session identifier, activities like "passing the remote" or mobile systems pose no problems. RTSP does not address the issue of how to arbitrate between multiple clients that want to control a single stream. In many cases, informal, social mechanisms coordinated, say, via the audio channel in a conference, will be sufficient. Indeed, it may be desirable to allow many participants to have potential control, so that, for example, any group member that wants to pause a video to ask a question can do so.

5.1 Caching

One of the differences in functionality between ftp and HTTP is the ability of HTTP to support caching of responses. For RTSP, caching of RTSP responses makes little sense, except possibly for presentation descriptions returned by DESCRIBE. However, it is clearly desirable to cache retrieved media-on-demand streams closer to the client. Given its derivation from HTTP, this turns out to be reasonably straightforward. A caching proxy listens in on the media stream passing through it towards the client and stores it locally. Subsequent requests are then served from the cache. Since the proxy cache has to buffer data in any event, it may decide to retrieve a presentation at the fastest speed that the network can support, using TCP, while delivering the cached material to the client using UDP. In many cases, this will result in better client and network performance. The normal expiration and validity-checking mechanisms employed by HTTP such as If-Modified-Since apply here as well.

HTTP/1.1 supports retrieval of ranges of data rather than the whole object referenced by the URL; this seems to be rarely used. For RTSP, the case of a proxy cache only holding time segments of a media stream is likely to be much more common. This means that a proxy cache may start serving a client from its local cache and then discover that the previous recipient skipped a piece of the presentation, now missing in the cache. The proxy cache then has to connect to the origin server and fill in the missing material, hopefully without introducing gaps at the client.

5.2 Combining SIP and RTSP

While SIP could also be used to initiate a pure media-on-demand session, this is not likely to be necessary in most cases. Note the difference between the conference invitation and initiating delivery of stored or live content. In the conference case, the calling party uses SIP to send the session description to the called party. The session description is likely to exist only for this call and is not stored on a server. The called party signals the call disposition and selects a suitable subset of the offered media types for communication. There is no concept of a media object with a long-term name.

For stored and live media, the presentation description, uniquely identified by an RTSP URL, is assumed to be available to the client through some means outside of RTSP, as discussed above. The client thus only has to refer to the URL when communicating with the server.

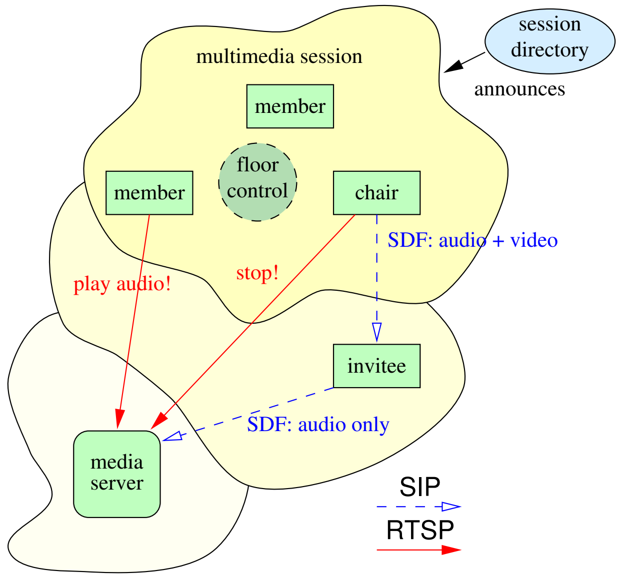

Figure 5: Internet conferencing example combining SIP and RTSP

For playing media into a conference, several methods are available. H.323 conferences, for example, require a new member to register with the RAS or an MCU. SIP-initiated conferences may require that participants encrypt data using a session key transferred as part of the session description. A media server cannot just play data to a given network address. In that case, the server has to "speak" the appropriate control protocol such as H.323 and is invited to the conference by a participant, just like a human participant. Any member of the conference can then initiate an RTSP session with the media server and direct the server to send media data to the conference identified by a Conference header field in the RTSP SETUP message. This assumes that conferences are named and that the server can associate stored media content with appropriate media in the conference session description. Note that the party inviting the server into the conference and the one controlling the playback or recording do not have to be the same (Fig. 5).

For Internet multicast conferences, it is often sufficient that the conference participant simply describes the network addresses used by the conference in an RTSP SETUP request, without formally inviting the server using SIP.

As mentioned earlier, RTSP offers the ability to "pass around the remote control" between conference participants. Participants have to know the session identifier and stream URL to control a media stream. SIP may be used to convey this information, either passing it on to the individual next in line to control the media stream or making the information generally available and relying on a floor control protocol to avoid conflicts.

6 RTSL: Description of Multimedia Presentations

Both SIP and RTSP need a data structure to describe the session or presentation they are initiating and controlling. The current session description protocol [3] works well when describing "live" multimedia sessions as found on the Mbone. However, it is not as well suited for describing stored sessions where a user can, for example, piece together a movie from different sound tracks and video versions. We have developed a simple hierarchical description called SDF, suitable for both SIP and RTSP. It describes presentations as a hierarchy of sequential, alternative and time-parallel streams. Each stream might well reside on a different server for load sharing or copyright reasons. The design of SDF found its way into a proposed SGML-based description called RTSL proposed by Progressive Networks which is currently under study by a working group of the World-Wide Web Consortium. An example is shown in Fig. 6, however, functionality and syntax are likely to change. (Using a language similar to HTML has the advantage that a backward-compatible page can be authored which only contains RTSP URLs within regular HTML. Also, search engines can index these pages without having to parse RTSL.) For simplicity, RTSL is intentionally purely descriptive and contains no scripting functionality.

7 Implementation

A client-side architecture is under development that supports stored, live and conferencing multimedia. An outline is shown in Fig. 7. The same media agents are used by all three applications. Each media agent has only a minimum user interface to achieve maximum reusability. Media agents and controllers communicate through a host-local conference control bus [39] implemented as either a central server or multicast with a time-to-live value of zero. This is a generalization of the conference bus employed by vat and vic.

The session directory shown in Fig. 7 listens to Mbone-style conference announcements formatted as SDP carried in SAP messages [40]. Rather than maintaining its own calendar, the session directory client communicates with the user's general calendar using a subset of SIP. Only the invitation message is needed here, indicated by the protocol designation SIP*. The calendar notes Mbone conferences along with the user's other appointments. Similarly, a user can enter a conference directly into the calendar. The session directory listens for these invitations and announces them to the world at large, after adding dynamically allocated multicast addresses to the description.

The SIP protocol is implemented for each logged-in user by the integrated session controller, *isc*. Since users may not always be logged on or may not have the conferencing tools running, and since there may be several users per host, a daemon receives incoming SIP requests on a well-known port and passes them on to the user-specific session controller. In the implementation described here, there is only a single session controller per user, showing all active sessions and their members in a single interface, saving screen real-estate. The session controller asks the owner whether she wants to accept an incoming call, while automated call handling is done by the daemon. A call may be rejected, forwarded or accepted based on the caller's identity, the urgency of the call or the subject matter indicated. However, an interface with the user's calendar offers much richer functionality. For example, a call could be automatically forwarded to the secretary when a time slot is blocked with a meeting. Depending on the identity of the caller and the privacy level indicated for an appointment, the caller may be provided with different levels of detail from "busy" to "in a meeting with Mark until 4 pm".

The functionality of the common resource reservation agent and the floor controller are discussed elsewhere [41].

8 Status and Future Work

We are currently completing an ISDN gateway through which POTS callers can reach those connected through SIP and RTP and vice versa [42]. This gateway uses a new call processing language, CPL, to handle both ISDN and SIP calls and can connect the two [43]. CPL combines the features of a Tcl-like scripting language with a state-based language to reflect the nature of ISDN calls as well as to simplify implementations of voice response systems. The system already supports remote participation in Mbone conferences, simply by selecting by touch tones from the menu of current sessions, read from the received SDP packets by text-to-speech software, or punching in a multicast address.

An extension of SIP to support MCUs and fully-meshed unicast conferences is required, as not all networks support network-layer multicast.

An initial RTSP server using a telnet client was implemented shortly after the first draft of the specification. Sev-

```
<title>Twister</title>
<session>
  <group language=en lipsync>
     <switch>
        <track type=audio
              e="PCMU/8000/1"
              src="rtsp://audio.example.com/twister/audio.en/lofi">
        <track type=audio
              e="DVI4/16000/2" pt="90 DVI4/8000/1"
              src="rtsp://audio.example.com/twister/audio.en/hifi">
        </switch>
     <track type="video/jpeg"
         src="rtspu://video.example.com/twister/video">
    </group>
</session>
```

Figure 6: Sample RTSP session description

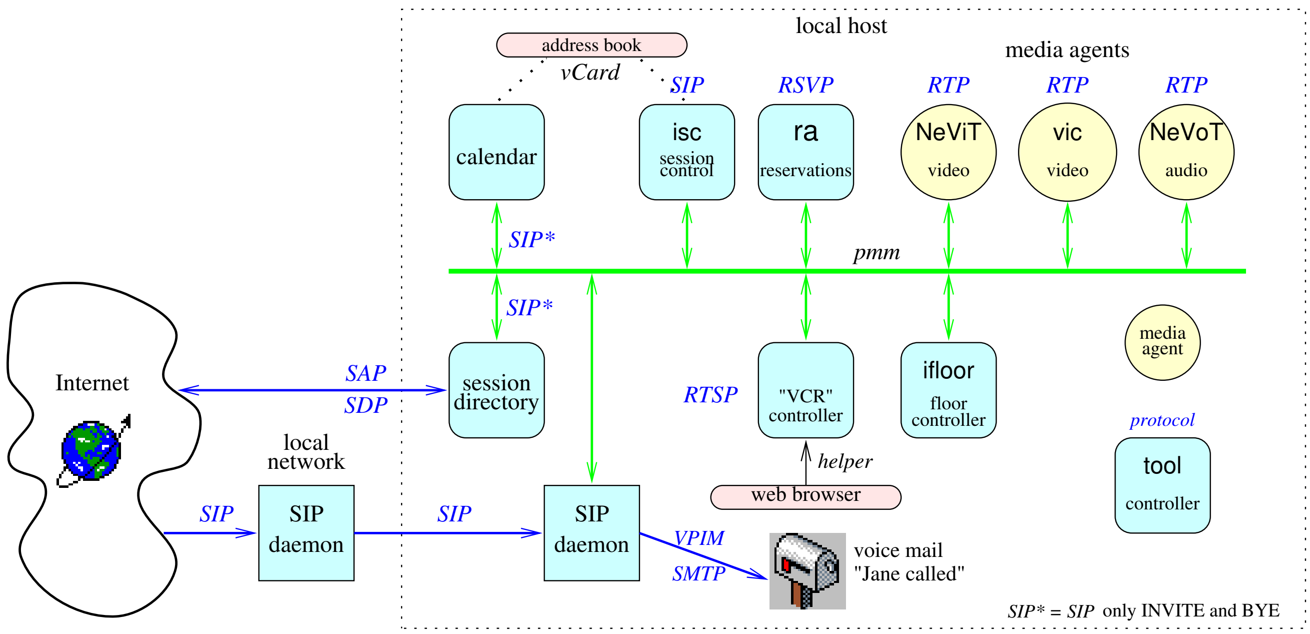

Figure 7: Possible client conferencing architecture using SIP and RTSP

eral freely available server and client implementations on different operating system platforms are expected to be available by the end of summer 1997, with standard Mbone tools as media clients.

We are currently studying how to offer near media-on-demand services where the server maintains a dynamic set of multicast groups and directs clients to the multicast group that is closest in time to the client's desired play point. For that, responses to the PLAY request must be able to change the client's multicast address.

The Internet multimedia architecture is still missing two pieces, namely a floor control protocol and a shared drawing protocol. (Shared applications are likely to be addressed by sharing either Windows or X.) The ITU T.120 series of protocols offers both, but offers opportunities for simplification and support of large multicast groups without a central server.

In the future, the author believes that the emphasis will shift from conferencing and stored media applications that are visible to the user as such to "embedded applications" that work behind the scenes of web pages, games and virtual reality. For example, approaching an object in VRML space may well trigger, via RTSP, delivery of a Foley sound.

9 Acknowledgments

The current version of SIP is also based on work of Mark Handley and Eve Schooler [44]. RTSP is based on the merger of a proposal by Anup Rao, Robert Lanphier et al. [45] and RTSP' [46] by the author.

The call processing language CPL has been developed by Frank Oertel [43]. An initial version of the Sun 'cm' calendar interface was developed by Yuwen Tu, Rui-Gang Yang and Chaomei Long. Stefan Hoffmann is implementing SIP in the daemon and *isc*.

The comments of Jonathan Rosenberg are gratefully acknowledged. Application development is partially funded by the Deutsches Forschungsnetz (DFN) through the US-MInt project.

References

[1] H. Schulzrinne, "RTP profile for audio and video conferences with minimal control," Request for Comments (Proposed Standard) RFC 1890, Internet Engineering Task Force, Jan. 1996.

[2] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new resource ReSerVation protocol," *IEEE Network*, vol. 7, pp. 8–18, Sept. 1993.

[3] M. Handley, "SDP: Session description protocol," Internet Draft, Internet Engineering Task Force, Nov. 1996. Work in progress.

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

[5] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," Internet Draft, Internet Engineering Task Force, Mar. 1997. Work in progress.

[6] D. T. Magill, "Adaptive speech compression for packet communication systems," in *Conference record of the IEEE National Telecommunications Conference*, pp. 29D–1 – 29D–5, 1973.

[7] Anonymous, "Special issue on packet switched voice and data communication," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, Dec. 1983.

[8] Arango *et al.*, "Touring machine system," *Communications ACM*, vol. 36, pp. 68–77, Jan. 1993.

[9] E. M. Schooler and S. L. Casner, "A packet-switched multimedia conferencing system," *SIGOIS (ACM Special Interest Group on Office Information Systems) Bulletin*, vol. 10, pp. 12–22, Jan. 1989.

[10] H. Eriksson, "MBONE: The multicast backbone," *Communications ACM*, vol. 37, pp. 54–60, Aug. 1994.

[11] V. Jacobson, "Multimedia conferencing on the Internet," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (London, England), Aug. 1994. Tutorial slides.

[12] R. Frederick, "Experiences with real-time software video compression," in *Sixth International Workshop on Packet Video*, (Portland, Oregon), Sept. 1994.

[13] 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.

[14] P. V. Rangan and D. C. Swinehart, "Software architecture for integration of video services in the Etherphone environment," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1395–1404, Dec. 1991.

[15] S. R. Ahuja and J. R. Ensor, "Call and connection management: making desktop conferencing systems a real service," *ACM Computer Communication Review*, vol. 22, pp. 10–11, Mar. 1992.

[16] E. M. Schooler, S. L. Casner, and J. Postel, "Multimedia conferencing: Has it come of age?," in *Proceedings of the 24th Hawaii International Conference on System Science*, vol. 3, (Hawaii), pp. 707–716, IEEE, Jan. 1991.

[17] E. Schooler and S. L. Casner, "An architecture for multimedia connection management," *ACM Computer Communication Review*, vol. 22, pp. 73–74, Mar. 1992.

[18] V. Balabanian, L. Casey, N. Greene, and C. Adams, "An introduction to digital storage media – command and control," *IEEE Communications Magazine*, vol. 34, pp. 122–127, Nov. 1996.

[19] International Telecommunication Union, "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1996.

[20] P. Lantz, "Usage of H.323 on the Internet," Internet Draft, Internet Engineering Task Force, Feb. 1997. Work in progress.

[21] C. Rigney, "RADIUS accounting," Request for Comments (Informational) RFC 2059, Internet Engineering Task Force, Jan. 1997.

[22] F. Yergeau, "UTF-8, a transformation format of unicode and ISO 10646," Request for Comments (Informational) RFC 2044, Internet Engineering Task Force, Oct. 1996.

[23] D. E. Eastlake, "Universal payment preamble," Internet Draft, Internet Engineering Task Force, Oct. 1996. Work in progress.

[24] J. Franks, P. Hallam-Baker, J. Hostetler, P. A. Luotonen, and E. L. Stewart, "An extension to HTTP: digest access authentication," Request for Comments (Proposed Standard) RFC 2069, Internet Engineering Task Force, Jan. 1997.

[25] T. Krauskopf, J. Miller, P. Resnick, and W. Treese, "PICS label distribution label syntax and communication protocols, version 1.1," W3C Recommendation REC-PICS-labels-961031, World Wide Web Consortium, Cambridge, Massachusetts, Oct. 1996.

[26] D. Connolly, "PEP: an extension mechanism for HTTP," Internet Draft, Internet Engineering Task Force, Jan. 1997. Work in progress.

[27] D. Kristol and L. Montulli, "HTTP state management mechanism," Request for Comments (Proposed Standard) RFC 2109, Internet Engineering Task Force, Feb. 1997.

[28] K. Holtman and A. Muntz, "Transparent Content Negotiation in HTTP," Internet Draft, Internet Engineering Task Force, Nov. 1997. Work in progress.

[29] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," Request for Comments (Proposed Standard) RFC 2068, Internet Engineering Task Force, Jan. 1997.

[30] P. Karn and C. Partridge, "Improving round-trip time estimates in reliable transport protocols," *ACM Computer Communication Review*, vol. 17, Aug. 1987. SIGCOMM '87 Workshop.

[31] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Request for Comments (Proposed Standard) RFC 1889, Internet Engineering Task Force, Jan. 1996.

[32] V. Jacobson, S. McCanne, and S. Floyd, "A conferencing architecture for light-weight sessions," Nov. 1993. MICE seminar series (transparencies).

[33] W. Yeong, T. Howes, and S. Kille, "Lightweight directory access protocol," Request for Comments (Draft Standard) RFC 1777, Internet Engineering Task Force, Mar. 1995. (Obsoletes RFC1487).

[34] A. Gulbrandsen and P. Vixie, "A DNS RR for specifying the location of services (DNS SRV)," Request for Comments (Experimental) RFC 2052, Internet Engineering Task Force, Oct. 1996.

[35] R. Pandya, "Emerging mobile and personal communication systems," *IEEE Communications Magazine*, vol. 33, pp. 44–52, June 1995.

[36] R. Williams, "User location service," Internet Draft, Internet Engineering Task Force, Feb. 1996. Work in progress.

[37] S. Petrack, "The call management agent system - requirements and architecture," Feb. 1997. Unpublished manuscript.

[38] S. Shenker, A. Weinrib, and E. Schooler, "Managing shared ephemeral state: Policy and mechanism," in *Proc. of the International Workshop on Multimedia Transport and Teleservices (COST237)*, (Vienna, Austria), Nov. 1994.

[39] 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), (Durham, New Hampshire), pp. 231–242, Springer, Apr. 1995.

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

[41] 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), IEEE Computer Society, Aug. 1996.

[42] C. Zahl, "Aufbau und Konfiguration von Internet-Multimedia-Konferenzen ber Verbindungen niedriger Bandbreite (setup and configuration of internet multimedia conferences using low-bandwidth links)," Master's thesis, Berlin University of Technology, Berlin, Germany, Feb. 1997. Diplomarbeit.

[43] F. Oertel, "Integration von ISDN-Teilnehmern in Internet-Multimedia-Konferenzen (integration of ISDN subscribers into internet multimedia conferences)," Master's thesis, Berlin University of Technology, Berlin, Germany, Feb. 1997. Diplomarbeit.

[44] M. Handley and E. Schooler, "Session invitation protocol," Internet Draft, Internet Engineering Task Force, Feb. 1996. Work in progress (expired).

[45] A. Rao and R. Lanphier, "Real time streaming protocol (RTSP)," Internet Draft, Internet Engineering Task Force, Nov. 1996. Work in progress.

[46] H. Schulzrinne, "A real-time stream control protocol (RTSP')," Internet Draft, Internet Engineering Task Force, Dec. 1996. Work in progress.