Signaling for Internet Telephony

Henning Schulzrinne
Columbia University
hgs@cs.columbia.edu
M/S 0401
1214 Amsterdam Avenue
New York, NY 10027

Jonathan Rosenberg
Bell Laboratories
jdrosen@bell-labs.com
Rm. 4C-526
101 Crawfords Corner Rd.
Holmdel, NJ 07733

January 31, 1998

Abstract

Internet telephony must offer the standard telephony services. However, the transition to Internet-based telephony services also provides an opportunity to create new services more rapidly and with lower complexity than in the existing public switched telephone network (PSTN). The Session Initiation Protocol (SIP) is a signaling protocol that creates, modifies and terminates associations between Internet end systems, including conferences and point-to-point calls. SIP supports unicast, mesh and multicast conferences, as well as combinations of these modes. SIP implements services such as call forwarding and transfer, placing calls on hold, camp-on and call queueing by a small set of call handling primitives. SIP implementations can re-use parts of other Internet service protocols such as HTTP and the Real-Time Stream Protocol (RTSP). In this paper, we describe SIP, and show how its basic primitives can be used to construct a wide range of telephony services.

1 Introduction

Internet telephony requires a range of protocols, ranging from those needed for transporting real-time data across the network, to quality-of-service-aware routing protocols, to resource reservation, QOS-aware network management and billing protocols. In addition, Internet telephony, defined here as synchronous voice or multimedia communication between two or more parties, requires a means for prospective communications partners to find each other and to signal to the other party their desire to communicate. We refer to this functionality as *Internet telephony signaling*. The need for signaling functionality distinguishes Internet telephony from other Internet multimedia services such as broadcast and media-on-demand services. In addition, IPtel signaling can be used to integrate different Internet multimedia modalities, as discussed in Section 6.

IPtel signaling as we understand it creates and manages *calls*. We define a call as a named association between applications that is explicitly set up and torn down. Examples of calls are two-party phone calls, a multimedia conference or a multi-player game. We do not require that calls have to have media streams associated with them, but this is likely to be the common case.

Unlike traditional telephony signaling, we make the signaling of calls independent of the notion of media "connections" or streams. Note that in contrast to systems such as H.323 [1], we do not require that all participants within a conference have to be aware of the existence of a call. Conversely, a call participant may not be generating media streams. For example, when a lecture is multicast, the initiator of the call may not be sending or receiving data.

We see Internet telephony signaling encompassing a number of functions: *Name translation and user location* involves the mapping between names of different levels of abstraction, e.g., a common name at a domain and a user name at a particular Internet host. These translations may involve simple table lookups at the server or may involve locating the party, as described in Section 4.2.

Feature negotiation allows a group of end systems to agree on what media to exchange and their respective parameters such as encodings. The set and type of media need not be uniform within a call, as different point-to-point sessions may involve different media and media parameters. Many software codecs are able to receive different encodings within a single conference and in parallel, for example, while being restricted to sending one type of media for each stream.

Any call participant can invite others into an existing call and terminate associations with some (*call participant management*). During the call, participants should be able to transfer and hold other users. The most general model of a multi-party association is that of a full or partial mesh of invitations, with the possible addition of multicast distribution between some or all participants.

Feature changes make it possible to adjust the composition of media sessions during the course of a call, either because the participants require additional or reduced functionality or because of constraints imposed or removed by the addition or removal of call participants.

Not all of these functions have to be addressed by one protocol. For example, H.323 may be used to establish sessions between the end system and the gateway, while the Session Initiation Protocol (SIP), the protocol described here, might be responsible for gateway-to-gateway signaling.

Other conference management functions are beyond the scope of signaling. These include distributed queue management for floor control and distributed counting for voting. Unlike signaling, both of these require some form of reliable multicast; however, for small groups, a multipoint controller can perform the replication. Signaling can be used to introduce this functionality into conferences as needed.

In Section 2, we discuss the basic architecture for telephony signaling services. In Section 3 we discuss the basics of SIP operation, its addressing structure, message syntax and transport. In Section 4, we discuss how SIP can be used for telephony services, focusing on how services are constructed from simple primitive tools. We then briefly discuss, in Section 6, the interaction of telephony signaling with stored media control protocols. We then mention related work in Section 7, and conclude in Section 8.

2 Internet Telephony Architecture

2.1 Separation of Signaling Functionality

Unlike circuit-switched telephony, Internet telephony services are built on a range of packet switched protocols, as illustrated in Fig. 1. For example, the functionality of the SS7 telephony signaling protocol encompasses routing, resource reservation, call admission, address translation, call establishment, call management and billing. In an Internet environment, routing is handled by protocols such as BGP [2], resource reservation by RSVP [3] or other resource reservation protocols [4]. SIP, described here, translates application-layer addresses, establishes and manages calls. There is currently no Internet telephony billing protocol in the Internet, although RADIUS [5], in combination with SIP authentication, may initially serve that purpose.

This separation of concerns affords greater architectural flexibility. For example, Internet telephony may be used without per-call resource reservation in networks with sufficient capacity; billing may not be necessary in a PBX-like environment. On the other hand, removing the "atomicity" of call setup found in the current telephone system also breaks assumptions: since call setup and resource reservation are distinct, one may succeed, while the other may fail. If resources are reserved first, the caller may incur a cost for holding those resources while "the phone rings", even though the call is not answered. (If the network does not charge for reservations that are not actually used, the network becomes vulnerable to denial-of-service

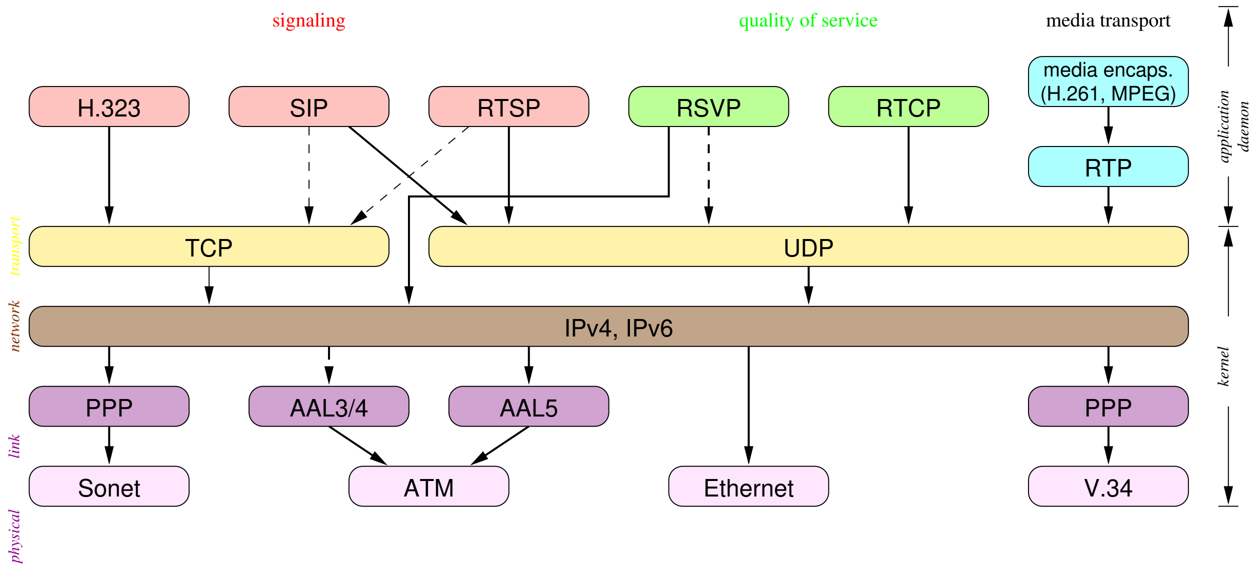

Figure 1: Protocol architecture for Internet multimedia services

attacks, where the attacker can block others from making reservations.) Also, in the phone system, the resource needs for a single leg of a call are known ahead of time; this is clearly not the case for Internet telephony calls, where the callee may choose to communicate with only a subset of the media offered by the caller.

2.2 Internet Multimedia Sessions

We treat a two-party phone call as a special, if important, case of conferencing. 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 through the upper-layer transport protocols, e.g., the RTP CNAME [6]. 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" [7]; unlike central-registry models, it scales to very large conferences and survives network partitions.

A potential participant in such a conference or Internet phone call has to somehow become aware of the existence of a session. We can distinguish two mechanisms: in the first, the potential participant actively seeks out potential sessions of interest, for example, by perusing a web page with conference listings, listening to a multicast group carrying session announcements or subscribing to a mailing list or newsgroup announcing conferences. All these mechanisms have in common that the creator of the session does not know if and when a particular person receives the description of the session. One could imagine sending email to the intended participants of a session. However, as it is typically delivered and read at intervals measured in minutes, email is unsuitable for the equivalent of a phone call. Email also does not offer a ready indication that the recipient has read the message and cannot easily indicate a willingness to participate in a conference.[1]

In current Internet multimedia applications using the light-weight session model, sessions and their attributes are announced using a well-known multicast address and a simple textual description (SDP) [8].

[1] SIP can use email as a last-resort delivery mechanism, which is useful for inviting users to long-running conferences.

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.

2.3 The Role of SIP

While we will use the term "Internet telephony" or "IP telephony" (IPtel) throughout the paper, it should be understood that all of the protocols mentioned are applicable not just to voice, but to general multimedia services, including video, text "chat", collaborative browsing, and application sharing. This is also true for SIP.

SIP is independent of the conference model and size. It works in the same manner whether calling a single party for a "classic" phone call, setting up a small conference or inviting another participant into an existing large multicast session with thousands of members. (In the ITU conferencing architecture, separate protocols, H.323 and H.332, respectively, are used.)

Besides the difference between the circuit-switched and packet-switched carriage of voice and other media, the public switched telephone system (PSTN) and IPtel as described here differ in a number of control aspects. ISDN, SS7 and SIP all separate the control path from the data path, but to differing degrees. ISDN signaling (Q.931) is closely associated with the data channel, in that they are carried in the same lower-layer multiplex. SS7 signaling is physically separate from the data path, but tied hop-by-hop, so that the signaling protocol traverses the same switching nodes as the voice traffic. SS7 traffic uses a different physical network, typically 64 kb/s redundant links between service control points. SIP, on the other hand, completely separates the control path: a SIP request may travel a completely different route from the data traffic, yet SIP requests use the same Internet infrastructure as the data.

SIP adds another separation of functionality, namely between call establishment and call description. The SIP requests described in Section 3 deal with a "call" as an association between two or more parties as a whole, without being concerned with what media constitutes the call. The makeup of the call is described using the session description contained in the request. SIP conveys the type of the description and allows server and client to negotiate acceptable description formats, but makes no other assumptions about the content. This allows session formats to evolve independently of the signaling protocol and allows SIP to be used in applications beyond IT.

Due to the limited signaling abilities of PSTN end systems, PSTN addresses (phone numbers) are overloaded with at least four functions: end point identification, service indication, indication of who pays for the call and carrier selection. The PSTN also ties call origination with payment, except as modified by the address (800 numbers). SIP addresses, in contrast, could incorporate these functionalities, but in general it is probably preferable to, for example, indicate carrier preference through name mapping and use authentication as a means to indicate willingness to pay.

Internet telephony signaling needs to be able to establish sessions between IP-connected end systems, between an end system and a gateway to another network such as the PSTN or an H.323-controlled system and finally between gateways. While separate protocols could be used for each of these, SIP tries to address all three modes.

Table 1 shows the possible combinations of unicast and multicast signaling and communications. For example, inviting somebody to a multicast conference (e.g., of the type found on the Mbone [9]) requires unicast signaling. For automatic call distribution (ACD), where a caller wants to reach the first available person, multicast signaling may be useful. Finally, one may want to invite groups of people to a multicast conference. The last mode differs in that, to avoid request implosion, invitees should not respond to the invitation, except possibly in a carefully rate-controlled way [10]. Multicast signaling also requires that potential invitees already expect an invitation, i.e., subscribe to the multicast group. Currently, the Mbone uses multicast announcements to distribute information about upcoming conferences and other multicast events [11]. It is possible to integrate conference announcements and invitation, although the semantics differ slightly: conference announcements are usually for some time in the future and do not generally alert the user, while invitations request immediate communications. However, the combination of SIP and SDP supports both immediate and future sessions.

signaling	conference	
	unicast	multicast
unicast	Internet telephony	conference invitation
multicast	ACD	group invitation

Table 1: Unicast and multicast signaling and conferences

3 The Session Initiation Protocol

3.1 Overview

SIP is a client-server protocol, with requests issued by the client and responses returned by the server. A single call may involve several servers and clients, as requests may be forwarded. This is similar to the HTTP model of clients, origin and proxy servers. Unlike HTTP, SIP servers do not cache data. SIP servers may cache call state or search results. A single host may well act as client and server for the same request, as discussed in Section 3.5.

As in HTTP, the client requests invoke *methods* on the server. Requests and responses are textual (see Section 3.7) and contain header fields which convey call properties and service information. SIP reuses many of the header fields used in HTTP, such as the entity headers (e.g., Content-type) and authentication headers. This allows for code reuse, and simplifies integration of SIP servers with web servers.

Calls in SIP have the following properties: The *logical call source* indicates the entity that is requesting the call (the originator). This may not be the entity that is actually sending the request, as proxies may send requests on behalf of other users. In SIP messages, this property is conveyed in the From header field. The *logical call destination* contained in the To field names the party who the originator wishes to contact (the recipient). The *media destination* conveys the location (IP address and port) where the media (audio, video, data) are to be sent for a particular recipient. This address may not be the same address as the logical call destination. Media destinations are conveyed as part of the payload of a SIP message. *Media capabilities* convey the media that a participant is capable of receiving and their attributes. Media capabilities and media destinations are conveyed jointly as part of the payload of a SIP message. Currently, the Session Description Protocol (SDP) [12] serves as the common format, although others are likely to find use in the future. SDP expresses lists of capabilities for audio and video and indicates where the media is to be sent to. It also allows to schedule media sessions into the future and schedule repeated sessions. Each call is logically identified by a globally (in time and space) unique *call identifier*, carried in the Call-ID field. The call identifier is created by the creator of the call and used by all call participants.

SIP defines several methods, described in detail below. The first three manage or prepare calls: INVITE invites a user to a conference, BYE terminates a connection between two users in a conference, OPTIONS solicits information about capabilities, but does not set up a call. STATUS informs another server about the progress of signaling actions that it has requested via the Also header (see below). ACK is used for reliable message exchanges for invitations. Finally, REGISTER conveys location information to a SIP server.

3.2 SIP Transport

SIP makes minimal assumptions about the underlying transport protocol. It can directly use any datagram or stream protocol, with the only restriction that a whole SIP request or response has to be either delivered in full or not at all. SIP can thus be used with UDP or TCP in the Internet, and with X.25, AAL5/ATM, CLNP, TP4, IPX or PPP elsewhere. Network addresses within SIP are also not restricted to being Internet addresses, but could be E.164 (PSTN) addresses, OSI addresses or private numbering plans. SIP also does not tie a session to the existence of a TCP connection, allowing to maintain calls even across reboots of some of the participants, as long as the end systems maintain call identifiers.

3.3 Addressing and Naming

To be invited and identified, the called party has 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"*, *"user@host"*, *"user@IP_address"* or *"phone-number@gateway"*. 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 name translation service. Addresses of the form *"phone-number@gateway"* designate PSTN phone numbers reachable via the named gateway.

SIP uses these addresses as part of SIP URLs, such as `sip://j.doe@example.com`. This URL may well be placed in a web page, so that clicking on the link initiates a call to that address, similar to a `mailto` [13] URL today.[2]

We anticipate that most users will be able to use their email address as their published SIP address. Email addresses already offer a basic location-independent form of addressing, in that the address does not have to designate a particular Internet host, but can be a domain, which is then resolved into one or more possible domain mail server hosts via DNS MX (mail exchange) records. This not only saves space on business cards, but also allows re-use of existing directory services such as LDAP [14], DNS MX records (as explained below) and email as a last-ditch means of delivering SIP invitations.

For email, finding the mail exchange host is often sufficient to deliver mail, as the user either logs in to the mail exchange host or uses protocols such as IMAP or POP to retrieve her mail. For interactive audio and video communications, however, participants are typically sending and receiving data on the workstation, PC or Internet appliance in their immediate physical proximity. Thus, SIP has to be able to resolve *"name@domain"* to *"user@host"*. 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. Also, hosts that connect via dial-up modems may acquire a different IP address each time.

3.4 Directory Services and SIP

The reader may wonder about the relationship between DNS, directory services such as LDAP and SIP. All can map an initial key to one or more values, such as network addresses. In this section, we describe how they compare and relate to SIP.

DNS offers a one-to-many mapping from a globally unique, hierarchical identifier to one or more host names or IP addresses. Lookups are by single (primary) key only, not generalized boolean queries across several parameters. Thus, because of the hierarchical nature and the restriction to primary keys only, DNS is not suited for looking up individual subscribers. DNS can, however, be used [15] by SIP to map the name of a domain to the name of a server for a specific request, further the published SIP address from the details of service provision.

[2] Most browsers currently do not allow to add handlers for non-HTTP protocols, unfortunately.

LDAP is a true directory service with search of attributes to a record. Given suitable schema, it answers questions like "what are the records matching a name Smith residing in New York City and older than 40 years". Such a database is appropriate to obtain a globally unique user "handle" from a set of attributes, but it appears likely that it will be used about as frequently as telephone directory services. For regular phone calls, it is far too cumbersome to provide sufficient information to obtain a small set of responses from the global set of available network addresses. SIP servers can make use of LDAP servers, however. For example, a SIP server may try to look up matches for "John.Smith" or "J.Smith" or "Smith" in a corporate database accessed via LDAP and return possible choices.

SIP can also provide a "programmable" directory service, where the response may depend on the identity of the caller, the type of media the caller wants to send or receive, the time of day, how many other calls the callee is currently engaged in, constellation of the stars or whatever else the server deems worthwhile. These dependencies are not easily cast into the framework of a relational database service, as the combination of attribute values is large, but sparse. Unlike a database, where updates tend to be infrequent and require action by an outside entity, a SIP response may yield different results with each call, without intervention of the callee. On the other hand, there is no real query language, but rather the *server* chooses which parameters of the call to interpret in handling the call. The call handling programs may well be stored in a database for large user populations.

3.5 Basic Operation

The most important SIP operation is that of inviting new participants to a call. A user first obtains an address where the user is to be called, of the form *name@domain*. The user then tries to translate this domain to an IP address where a server may be found. This translation is done by trying, in sequence, DNS SRV records [15] of type sip.udp and sip.tcp, MX, CNAME and finally A records. Once the server's IP address has been found, the user sends it an INVITE message using either UDP or TCP.

The server which receives the message is not likely to be the host where the user is actually located. Because of this, we define three different server types: proxy, redirect and user agent. A *proxy server* receives a request and then forwards the request towards the current location of the callee. For example, the server responsible for `example.com` may forward the call for `john.doe@example.com` to `doe@sales.example.com`. A Via header traces the progress of the invitation from server to server, allows responses to find their way back and helps servers to detect loops. A *redirect server* receives a request and informs the caller of the next hop server. The caller then contacts the next-hop server directly. Finally, a *user agent server* resides on the host where the user is situated. It is capable of querying the user about what to do with the call: accept, reject, or forward. Figures 3.5 and 3.5 show the behavior of SIP redirect and proxy servers, respectively.

Proxy servers can forward the invitation to multiple servers at once, in the hopes of contacting the user at one of the locations. They can also forward the invitation to multicast groups, effectively contacting multiple next hops in the most efficient manner.

Once the user agent server has been contacted, it sends a response back to the client. The response has a response code and response message. The codes fall into classes 100 through 600, similar to HTTP.

Unlike other requests, invitations cannot be answered immediately, as locating the callee and waiting for a human to answer may take several seconds. Calls may also be queued, e.g., if the callee is busy. Responses of the 100 class (denoted as 1xx) indicate call progress; they are always followed by other responses indicating the final outcome of the request.

While the 1xx responses are provisional, the other classes indicate the final status of the request: 2xx for success, 3xx for redirection, 4xx, 5xx and 6xx for client, server and global failures, respectively. 3xx responses list in a Location header alternate places where the user might be contacted. The response is always sent to the entity which sent the message to the server, not the originator of the request. To ensure

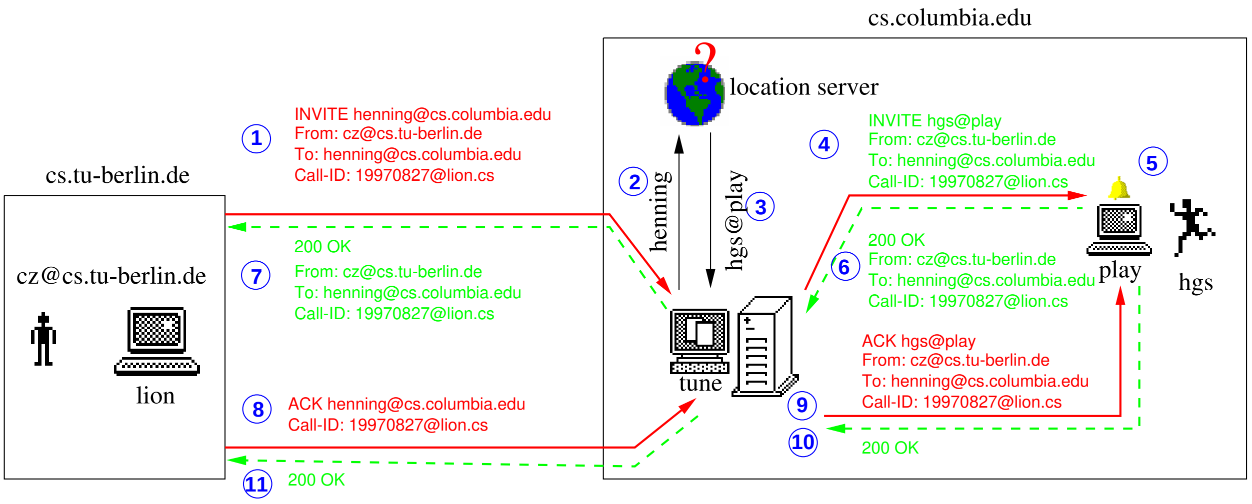

Figure 2: SIP invitation in proxy mode

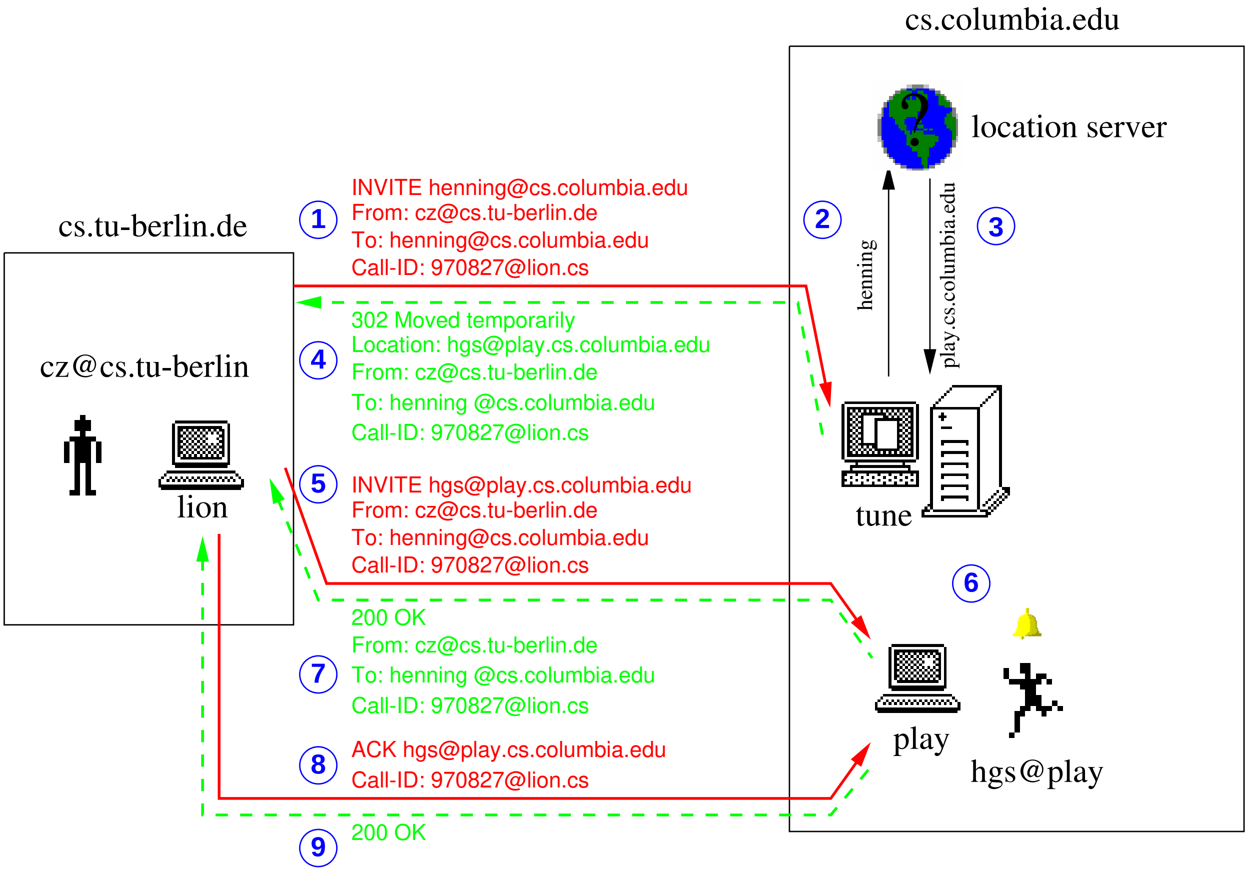

Figure 3: SIP invitation in redirect mode

reliability even with unreliable transport protocols, the server retransmits final responses until the client confirms receipt by sending an ACK request to the server.

All responses can include more detailed information. For example, a call to the central "switchboard" address may return a web page that includes links to the various departments in the company, providing navigation more appropriate to the Internet than an interactive voice response system (IVR).

3.6 SIP Message Reliability

As mentioned earlier, SIP makes minimal assumptions on the underlying transport protocol. Also, SIP operation is the same whether the transport protocol is reliable or not. Reliability is achieved by having the client retransmit requests every 0.5 seconds until either a progress report (1xx) or final status (≥ 200) response has been received.[3] It is expected that the server responds immediately with a progress report or possibly an error indication upon receipt of an invitation, but then may take a considerable amount to time until returning a final status, e.g., if it is ringing the phone. The server retransmits responses until the client acknowledges its receipt with an ACK. In effect, this amounts to a three-way handshake similar to setting up a TCP connection, requiring a total of 1.5 round-trip times and 3 messages to set up a call if no packets are lost. Pending invitations can be cancelled with a BYE request. The client and server state machines are shown in Fig. 3.6 and 3.6, respectively.

There is no "acknowledgement of acknowledgement" problem, since any response to the ACK has no significance to either side. It can be sent for reasons of symmetry with other requests, but is not required. The server simply retransmits the original final status response until an ACK is received. The client retransmits and ACK for every final status message.

Using application-layer reliability rather than TCP has the advantage that timers can be adjusted according to the requirements of a signaling application rather than being at the mercy of a kernel protocol stack. For example, standard TCP retransmits SYN packets after 6 and 24 packets, far too long for reasonable "post-dial delay" should a packet be lost. TCP features such as a flow- and congestion-controlled reliable byte stream are not particularly helpful, as SIP messages are likely to be short and sporadic. Unlike TCP, the problem with sequence number re-use is avoided by assigning each call an identifier which is unique across all hosts and never re-used in time. Servers are required to support both UDP and TCP, while a thin client such as a standalone "Internet telephone" may only support UDP.

Protocol operation is simplified by making each request idempotent; the recipient of the request simply checks if it has already executed the required action. For example, during an invitation, if the caller changes her mind as to details of the request, the latest request will be used.

An earlier version of SIP used a simple retransmission scheme, where requests would be re-sent periodically until a final response arrived, with no retransmission of responses. However, for invitations, this is undesirable for reasons of responsiveness and network load. If requests are widely spaced, the callee has no way of knowing quickly when a call attempt has been aborted. If responses are not retransmitted, the callee may have picked up the call and expects to find a media connection, but the caller does not find out until the next retransmission of the invitation. The caller's application would still be generating ring-back. If retransmissions are spaced closely enough to eliminate this uncertainty (with a retransmission interval of around a second, say), a single call attempt may generate dozens of invitation packets, increasing network traffic and impacting all servers in the path. Just retransmitting responses without ACK adds additional network overhead, since the callee has no way of knowing when it can stop retransmitting responses. The simple retransmission also makes implementation of long-lived connection attempts such as call queueing (see Section 4.1) very inefficient and difficult.

The ACK request primarily avoids unnecessarily retransmitting responses, but also eliminates the race condition where the caller gives up on a call just as the callee picks up the phone. The callee would not enable the transmission of data until the ACK has been received.

When interoperating with Q.931 ISDN signaling, the ACK request corresponds to the "Connect Ack" message.

[3] Most network paths suitable for interactive voice conversation will have delays smaller than this value.

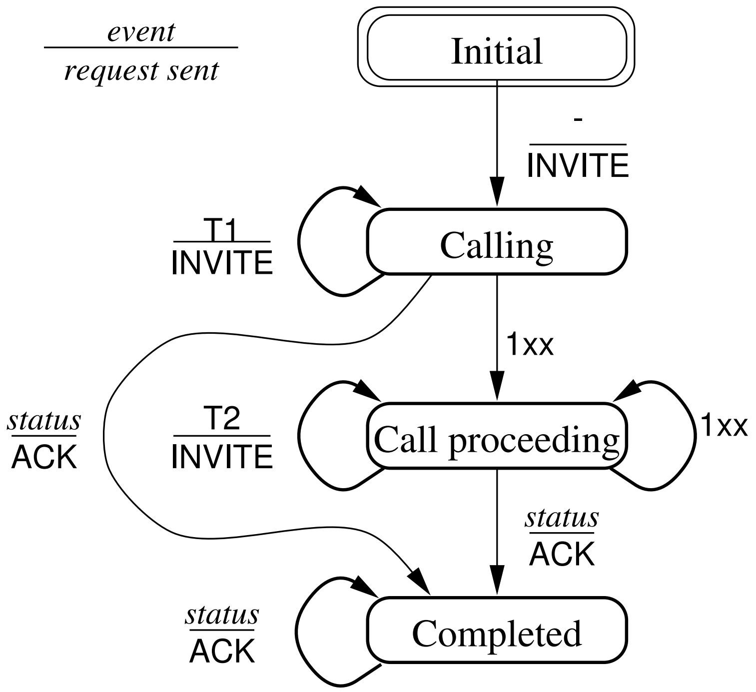

Figure 4: SIP state machine for client

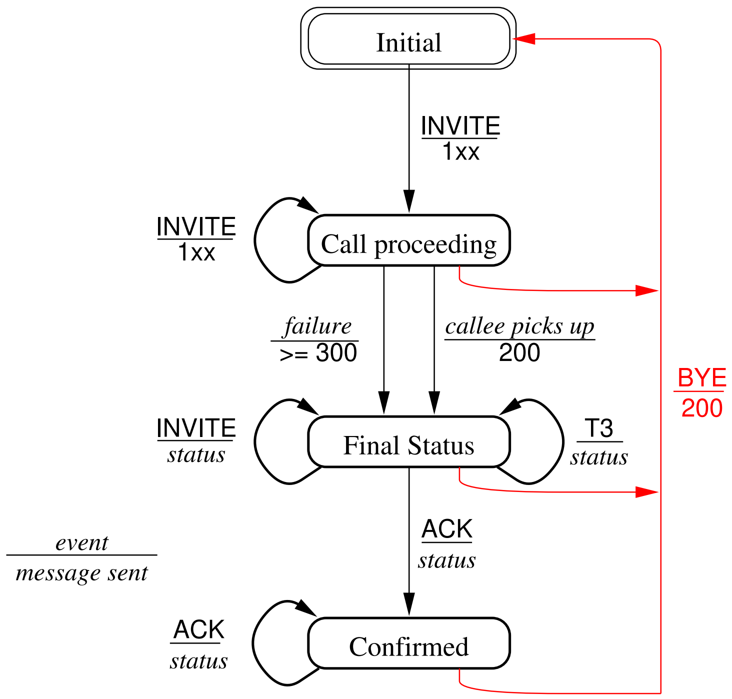

Figure 5: SIP state machine for server

3.7 SIP Message Syntax

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 [6], RSVP [3] and the RADIUS accounting protocol [16] 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 (BER), which have the advantage of describing the data *types* (not names) of data elements, are fairly verbose. Parsing of ASN.1 is cumbersome, as it requires a parser for both the description and the binary format. Tools for parsing PER are rare and cost around $30,000. 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. ASN.1 definitions are usually extensible in a backward-compatible way, but only by updating the central description. If two vendors add a new, optional field to BER or PER encodings without coordination, applications will get confused. For headers, the name of the header will often provide a hint as to its meaning. Many of the currently used email headers, for example, have evolved through such individual additions.

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 [17]. (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 [18], authentication [19], content labels and client-side access control [20], protocol extensions [21], state management [22] and content negotiation [23]. 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.

3.8 Protocol Extensions

Since IPtel is still immature, it is likely that additional signaling capabilities will be needed in the future. Also, individual implementations and vendors may want to add additional features. SIP is designed so that the client can either inquire about server abilities first or proceed under the assumption that the server supports the extension and then "back off" if the assumption was wrong.

Methods: As in HTTP, additional methods can be introduced. The server signals an error if a method requested by a client is not supported and informs it with the Public and Allow response headers about the methods that it does support. The OPTIONS request also returns the list of available methods.

Request and response headers: As in HTTP or SMTP, client and server can add request and response headers which are not crucial to interpreting the request or response without explicit indication. The entity receiving the header simply silently ignores headers that it does not understand. However, this mechanism is not sufficient, as it does not allow the client to include headers that are vital to interpreting the request. Rather than enumerating "need-to-know" non-standard headers, the SIP Require header indicates those features that the client needs from the server. The server must refuse the request if it does not understand one of the features enumerated. Feature names are either registered with the Internet Assigned Number Authority (IANA) or derived hierarchically from the feature owner's Internet domain name, giving hints as to where further information might be found. SIP uses this to ascertain whether telephony call-control functions are supported, avoiding the problem of partial implementations that have unpredictable sets of optional features.

Status codes: Status codes returned in responses are classified by their most-significant digit, so that the client knows whether the request was successful, failed temporarily or permanently. A textual status message offers a fall-back mechanism that allows the server to provide further human-readable information.

3.9 Security

Signaling requires security, primarily authentication, to prevent spoofing of calls, denial-of-service attacks and the equivalent of commercial unsolicited email ("spam"). This is particularly important since the traditional means of ensuring some privacy by unlisting telephone numbers is not directly applicable to IPtel signaling as long as email addresses are (semi) public.

SIP can make use of standard HTTP authentication mechanisms including basic (that is, clear-text password) and digest (challenge-response). Since TCP-based security protocols such as TLS [24] are not directly applicable to SIP if run over UDP, we are investigating whether cryptographically signed requests, e.g., using a variant of PGP or S/MIME, might add additional security.

For telephony, third-party signing of requests may be particularly useful. While a callee is unlikely to recognize, say, an individual employee at his local bank, having the call signed as originating from within that organization provides him with an additional means to filter and process calls.

We are currently investigating whether SIP and SDP should be extended to also support the generation of keys for encrypting media streams, e.g., using Diffie-Hellman key exchange.

3.10 Transport Issues

Signaling protocols such as H.323/H.225.0 and ISUP assume a reliable transport mechanism. SIP, on the other hand, may use any transport protocol that offers either reliable byte stream, reliable or unreliable datagram service, including TCP, RDP [25], UDP, IPX, or AAL5. While assuming a reliable transport protocol would simplify the SIP state machine somewhat, using TCP significantly increases call setup delay. First, the three-way handshake adds 1.5 round trip times of delay. Also, current TCP implementations are very conservative in their retransmission of the initial SYN packet, with retransmit delays of 6 and 24 seconds. Thus, even a single packet loss can lead to unacceptable call setup delays. Since SIP is primarily of interest for networks whose delay is low enough to allow interactive communications, we set the retransmit timeout to 500 ms. Finally, for some signaling operation such as searching, it is desirable to multicast SIP requests, making UDP preferable.

TCP features such as flow and congestion control are not particularly helpful for this application. However, for firewalls and transport-layer security protocol such as TLS [26], use of TCP may be required, so SIP allows both.

Even if TCP is used, SIP does not tie the existence of a call to that of the signaling connection. In contrast, H.323 assumes that the call is terminated when the H.245 [27] connection is closed. Avoiding relying on a TCP connection to maintain a call also simplifies hand-off in application-layer mobility.

4 Services

The model for development of telephony services (such as multi-party calls, call transfer, hold, etc.) using SIP is substantially different from other telephony architectures, such as H.323 and Q.931. These differences stem from the need for extensibility and growth. The Internet has thrived because it enabled rapid development and deployment of new applications without centralized control or "forklift upgrades" of all systems involved. Similarly, we would like the protocols that provide Internet telephony services to allow for quick development and deployment of new services. Furthermore, we would like these services to be available to existing endpoints, if possible.

To realize this goal, we have defined a set of tools that a SIP client has at its disposal in order to construct services. The behavior of a server in response to the invocation of these tools is also well defined. This allows clients to construct services by applying particular tools in a certain order.

The tools fall into two categories. The first are request methods. There are three request types which can be used as tools for creating services: INVITE, BYE, and OPTIONS. The other SIP request methods are not used for creating call services directly. The second type of tool are the header fields, primarily Call-Disposition, Also, Location, and Replaces. Each header field causes the server to perform a well-defined operation. Server behavior in response to receiving any combination of these three are also well defined. There are other header fields, of course, but these are either independent of those used for call services (the authentication headers being an example), or provide additional information which may be needed for call services (Call-ID, for example).

The header fields and message types are orthogonal. That is, the semantics defined for the four call service header fields are independent of which request or response message they are present in. This both simplifies implementation and allows for richer service offerings.

The next section discusses these tools and their operation. The sections which follow show how these tools can be used to construct a variety of advanced telephony services. For brevity, "A invites (drops) B

with Also: C" means that party A sends an INVITE (BYE) request to party B, with the Also header value of C. "A accepts" indicates that A returns a response of 200 OK.

4.1 Primitives

The tools described here essentially allow for constructing and destroying pieces of a *call mesh*, where this mesh represents the endpoints involved in the call (which may be users, bridges, media players, or any other relevant device), and the branches represent the logical connections which have been established between them.

The message tools are INVITE, OPTIONS, and BYE. Each of these request messages are sent from a client to a server. The server behavior in each of the three cases is simple.

The INVITE request indicates that client wishes to establish communication, or change some aspect of the communication, with the server. The server knows which is the case based on the Call-ID field. If the Call-ID field in the message is new (that is, the server has no other calls with that Call-ID), the call is new. If the Call-ID is not new, and the originator of the request is already in the call, the message is either a duplicate (known by the Cseq field, which is a simple sequence number), or contains an update about the call. An update is usually silently executed by the server, without informing the user, as the user has already accepted the call. If the Call-ID is not new, but the originator of the request is not in the call, then this is a new party being added to the existing call.

The BYE message indicates that the client wishes to terminate communication with the server. The BYE message must contain a Call-ID which is already active with the originator of the request.

The OPTIONS message is a null operation. It does not establish or tear down a call between client and server. However, several things do happen. First, the server returns an OK response to the client containing SDP which describes its capabilities. Secondly, the server will execute the actions specified by the header fields in the message. For example, an OPTIONS message can contain an Also field.

In conjunction with these messages, the header fields provide tools for additional services. Perhaps the most powerful of these is the Also header. This header contains a URL (generally a SIP URL, but not necessarily), which contains another entity that the server should call (by sending an INVITE to that URL). There may be many Also header fields, in which case the server should send an INVITE to all. When the server (now acting as a client) sends INVITE messages in response to an Also, it uses the same Call-ID from the original INVITE. Furthermore, the server will insert the Requested-By field into the invitation, containing the SIP URL of the client who sent the original request. This allows for some advanced services, some of which will be described below. The Also header can be present in any query.

As authoritative responses from the Also spawned invitations come back, the server should send provisional responses back to the client, indicating the result of those invitations. When an authoritative response has been received from all entities listed in Also fields, the server sends an authoritative response back to the client, indicating the number of parties in the Also fields which were finally connected. Since the informational responses may delay the transmission of the authoritative response, the client can specify whether the server should send the informational responses, or just send an authoritative response indicating the status of the client server connection. This service is specified with the Call-Disposition header.

The Location header is similar in function to the Also header when present in a request, except that it indicates alternatives to be tried until the first success. Thus, when there is only one Location header in the query, its function is identical to Also. When there are multiple Location headers in the request or response, the recipient chooses one of the URLs, and sends it an INVITE. The Location addresses may refer to different modes of communicating with the same person, or to different people. Since the Location header involves a choice, it can include parameters for providing the recipient with information in order to make that choice. These parameters include callee preference, priority, mode of communication (e.g., fax, pager, PSTN or Internet telephony) language spoken, whether the URL is for a mobile user, or whether the URL is for home or business.

The Replaces field is best described by analogy – Also is to INVITE as Replaces is to BYE. When present, the Replaces header indicates that the user should send a BYE to the parties indicated. It is allowed for the Replaces header to include a * as the URL. This indicates that this connection replaces all other connections with other parties with the same Call-ID.

In any message, there must be only Also header fields or Location header fields. There may also be Replaces header fields. In that case, the recipient of the message should first execute the Also or Location invitations. Then, when authoritative responses have been received, the Replaces is executed. This ordering can be reversed by so indicating in the Call-Disposition header.

The Call-Disposition header has been mentioned repeatedly. It is a powerful mechanism for expressing client preferences about call handling. The client can indicate that the call should not be forwarded ("do-not-forward"). It may also request that the call should be queued if the callee is busy ("queue"), implementing a type of camp-on services. A disposition of "status" asks the server to send back informational responses about the status of the Also invitations. New values can be added as needed, with a well-defined mechanism (see Section 3.8) to ensure that the client requests are understood.

The orthogonality of the functions provided by these headers and messages achieves two goals simultaneously. First, it simplifies processing, since complex relationships need not be tested and. they can be executed one at a time. Second, the orthogonality causes an exponential increase in the number of possible services which can be created by including these fields.

Using these logical building blocks, we can construct a wide variety of services. The following sections discuss some of the possibilities for forwarding, user location, transfer and conferencing services, but are by no means exhaustive.

4.2 Forwarding and User Location Services

The telephone network defines a range of forwarding and number translation services for different conditions, such as *call forwarding busy*, *call forwarding no response*, and *selective call forwarding*. 800 and 900-number services are also examples of such services. SIP generalizes forwarding to these and any other observable condition, result of a user location query or user preference. User preference can be expressed as rules or manual response to a call, e.g., by clicking on a "do not disturb" button when a call arrives.

The forwarding location can be determined in a number of ways. First, a SIP user agent can let the SIP server of a domain know of its presence via a REGISTER message. Other methods include the finger protocol [28], database accesses, for example PSTN Intelligent Network databases, or a query multicast on a local network.

All call-forwarding functions are usually instantiated with the Location header field, sent in the 300-class response to the original INVITE message. The Location field contains the possible destinations where the call should be forwarded to. The callee's user agent server or a server can send a redirection response based on any number of reasons, such as the caller, the time of day or availability of callee. When the decision to send a redirect response is made at the user agent end, the decision logic can be programmed in any desired way, with or without user interaction. When the decision is made by a redirect server, the decision can be made based on local policy at the redirect server, and by user specified preferences. We are proposing that user specified preferences are indicated by uploading a simple call processing directive function to the SIP server. This upload is accomplished by using the SIP REGISTER message. As SIP messages can contain any MIME type, the REGISTER message contains the directive, expressed as some script. When a call arrives for that user, the SIP server executes the script to arrive at a decision. This is much like the operation of the PSTN Intelligent Network, except the features are exposed to clients in a simple and scalable manner.

The caller invites the addresses listed in the Location headers. The Location header, as mentioned in Section 4, contains additional fields which can help the caller decide which address to use. These decisions can either be automated, or made through user interaction.

Since Location URLs can contain any URL, not just SIP URLs, calls can be forwarded between communication domains, for example, to a regular PSTN phone number, a web page for further information, an RTSP URL [29] for an answering machine or a mailto URL to leave an email.

The redirect mechanisms described here, when used several times, enable a host of forwarding functions which can range from simple forwards to personal mobility, i.e., the ability of a callee to be reached under one address regardless of the terminal being used). As an example, subscriber Alice may maintain a permanent, life-time "phone number" with a professional organization, say "alice@ieee.org". When she changed jobs, she notifies that organization to forward her calls to, say, "alice@employer.com". The SIP server at `employer.com` has access to the personnel database and forwards calls to Alice's department. Alice, in turn, programs her PC to forward calls to her wireless laptop she takes to classes at Columbia University, with the address `alice@cs.columbia.edu`. If she is currently disconnected, one of the alternatives offered in a Location header may be `pager://1-800-BEEPER?PIN=12345`.

There may be cases where a user wishes to find out where another user is actually located, and what media they can understand there, without actually making a call. This *user location service* can be invoked by sending an OPTIONS message to the user, instead of an INVITE. The same set of forwarding and redirect functions are available, but no call is actually set up.

5 Buddy Lists

Buddy lists is the term used to describe software that lets users be aware when friends are logged-on and available for communications. It had its origin in consumer on-line services like America Online, where "instant messaging" and textual chat are among the more popular services. Currently, there are products including ICQ ("I seek you") that perform similar functionality in the Internet. A primitive form of a buddy list is also found in the current Microsoft NetMeeting conferencing software, where a so-called ILS server records who is on-line and using the software [30].

Buddy lists are primarily suited for environments such as dial-up Internet services where people are only sporadically available through the network. However, in a more general sense, they can also serve as an indication of the willingness to communicate with different sets of people. People may prefer such indication to being interrupted by phone calls during a meeting. The SIP queueing mechanism already offers a form of communication awareness, of the form "I want to talk to you as soon as you are available". However, there are other cases where a colleague may want to be notified of another's presence, and then decide whether to establish contact. Also, it may be desirable to tie together the presence of several people; one might, for example, want to meet electronically when five out of the group of eight have come in in the morning, including the group leader.

In local environments, a number of CSCW systems providing awareness of other's people presence have been developed For example, Gaver [31] describes a system where connecting to and disconnecting from somebody's camera generates the sounds of a door creaking open and shutting. The Montage system [32] present the hallway model of awareness, with reciprocal and gradual video awareness.

While these awareness mechanisms are observer-triggered and require explicit action by the person wanting to "visit" somebody, buddy lists are triggered by actions of the person being waited for. A person lets it be known that he or she is available for communications, and some number of other people are notified of this fact more or less intrusively. Buddy systems like ICQ allow participants to put up "do not disturb" signs, require notification when somebody is watching or even require approval to be watched by a particular person.

The willingness to communicate can be detected either automatically or manually. Examples of automatic detection include computer keyboard activity, passive infrared (PIR) occupancy sensors, chair pressure mats or the presence of speech.

In SIP-based systems, the REGISTER request provides a basic mechanism to indicate presence. Just like calls, registrations are identified by a Call-ID. Two new request, HERE and GONE, request updates about the coming and goings of a particular individual, respectively. They both have the same reliability mechanism as INVITE, i.e., using 1xx informational responses and a final response acknowledged by ACK. Registrations and notifications are cancelled using BYE and expire automatically after a certain time enforced by the server.

Say, Alice wants to be notified when Bob is available to talk. Alice issues a "HERE Bob" request to Bob's home server. Bob's server initially answers with "100 Trying". If Bob is already registered, the server responds with "200 OK". If not, the server will delay that response until Bob shows up. Once Alice's client has received the "200 OK" response, it now wants to monitor when Bob leaves again, so it automatically issues a "GONE Bob" request.

The call processing language included in the registration can indicate how new notification requests should be handled, e.g., by disallowing them altogether, allowing them if notification is mutual, generating an email notification, (web-based) approval or by forwarding it to some other server. Forwarding would allow somebody to watch, say, `member@ieee.org`, yet have the actual matching be done by a server closer to the user's current location. In all likelihood, WATCH requests would not be forwarded to the user's terminal, but stop at a user-designated constant "home location register".

One problem with these buddy list mechanism is the scaling of registration and notification for large domains. For example, all 10 million members of America Online have the same domain name, aol.com. There are at least two ways to to address this. First, registrations and observations could all go to a single server, that then redirects both according to some private algorithm. Another alternative would be to embed scaling into the name resolution: A client would count the number of DNS entries with equal priority for the SIP registration server, say n, compute a hash h across the user name and then pick the $h \pmod n$th server.

5.1 Group Invitations

Existing telephony signaling protocols typically only support the invitation of a single individual. SIP aims to also allow calls that reach either the first available individual from a group, similar to automatic call distribution (ACD), or a whole group of callees, e.g., the whole department. To simplify error reporting and the client state machine, we limit each call to reaching a single individual, either through unicast, described in this section, or multicast, described later in Section 5.3.4. (There is one exception, namely using SIP to distribute announcements of scheduled events, where receivers are prohibited from generating responses.)

We refer to the calling of the first available individual as "reach-first" and to inviting a whole group as "reach-all". In both modes, the destination user name is actually the name of a list of individual addresses, to be resolved by the server. Examples of such names might be `sales@acme.com` or `ietf@ietf.org`. Just as for email addresses, there is no way that a user can know for certain whether an address is that of a list or of an individual. Since email is a distribution service, email only knows the equivalent of "reach-all", while for telephony both services make sense. By default, a caller expects a single individual to answer a call; reaching a group requires explicit setting of a flag in the Call-Disposition header. We first describe the reach-all service in Section 5.1.1 and then the reach-first service in Section 5.1.2.

5.1.1 Reach All

The caller declares its intention to set up a group call by setting the all flag in the Call-Disposition argument in its invitation. Reach-all service is then implemented in two stages. First, the caller sends an invitation to the named server. The server returns a "Multiple Choices" (300) response and enumerates the members of the list in Also headers. (Note that Location headers refer to multiple possible locations of a single individual, while Also headers list distinct individuals.) Employing the normal behavior for the Also header, the caller simply invites each listed individual, but with the same call identifier. It is up to the caller to decide whether to set up a mesh or create a multicast group. For the mesh case, the caller includes Also with the group address in the invitation, so that every group member goes through the same resolution process. (Since all will use the same call id, a callee that has rejected an earlier invitation from another group member will not be bothered again.)

5.1.2 Reach First

Reach-first appears not only in ACD systems, but is also useful for residential services such as having a single number ring at two different locations. Some phone companies offer this service when customers move from one residence to another; it is also useful for professionals that may work both at home and at an office or for a common phone number in a main and vacation residence.

5.2 Call Transfer Services

There are a number of different instances of transfer services possible, depending on whether the new connection is established before or after the disconnection and whether the initiator of the transfer is kept informed as to its progress. In all cases, it is immaterial as whether the caller or callee of the original call initiates the transfer.

The simplest type is *blind transfer*. In this scenario, user A is in a conversation with a set of users B_i, $i \in 1 \ldots n$. A would like to disconnect from the call and ask user C to connect with all B_i's instead. The transfer is called "blind" since User A does not need any confirmation of whether the transfer to C has succeeded. To implement this service, A drops all B_i's with C named in either the Location or Also header. Prompted by this header, all B_i's invite C as normal. Except for the Requested-By header indicating A, this looks like a normal call to C. If it is desired, A can ask to be informed about the status of the call by setting the Call-Disposition flag. However, these transfers are limited in that even if the transfer is not successful, the originating party is still disconnected.

An alternative transfer service which does not suffer this problem is possible, using a "make-before-break" approach. Here, the party initiating the transfer asks the other original call party to call the transfer destination. For example, customer C is in a call with secretary S. S wishes to transfer C to B_1 or B_2, should B_1 not be available. To do this, S invites C, with the same Call-ID as their existing call, "Also: B_1" and "Call-Disposition: status". This will cause C to invite B_1, with the "Requested-By: S". If B_1 is a busy, this response is echoed to S. Now, S invites C again, this time with "Also: B_2". Once S finds out the call $C - B_2$ has been set up, it can now drop C. Note that neither B_i nor C needs to know anything about this alternate transfer service.

In the *operator-assisted call transfer* service, the transferring user (say, a secretary S), wants to confer with the transfer recipient (here, boss B) to confirm that the transfer of the caller (customer C) is acceptable. Also, it wants B to initiate the call to C. The secretary can then either leave or stay in the call. The customer C invites the secretary S, who in turn initiates a new call to B asking for permission. If granted, S invites B again, this time with "Also: A", the Call-ID of the C-S call and "Call-Disposition: status". The secretary may then leave the conference by dropping C and B, may terminate the call with C, but remain connected to the boss, or may remain in the call, which has now become a three-way call. Compared to traditional telephony, all parties have a much clearer picture as to who is currently participating and what the status of the call transfer is.

The *auto-dialer* service is a variation on the operator-assisted call transfer. Here, the auto-dialer A takes the place of the secretary above and calls a potential customer C (usually at dinner). After the customer has answered, A then calls a third user, the telemarketer T. Upon success, A wants to connect T and C and drop out of the call.

This service is also easily implemented (in many ways, in fact) with the above mechanisms. The auto-dialer first invites C (Fig. 5.2). If C accepts, the auto-dialer invites the telemarketer with Also: C, and "Call-Disposition: status". The telemarketer then calls C, using the same Call-ID as for the connection between C and A. This causes the customer's application to treat this is a new party in the call, so it is accepted. The telemarker then indicates to the autodialer that the call has been completed. This causes the auto-dialer to drop the customer and the telemarketer. T and C are now connected.

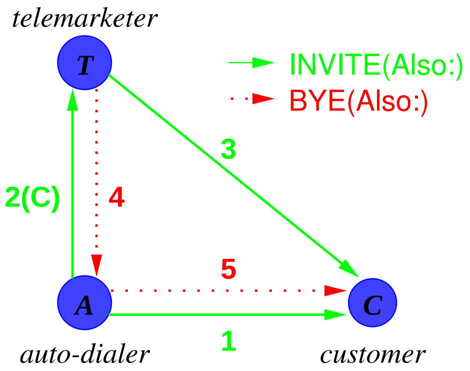

Figure 6: Auto-dialer (telemarketing) service

There are other ways to implement this service. The previous approach has the drawback that it causes the customer to first "see" the autodialer, then the telemarketer and the autodialer, and then just the telemarketer. This transition can be made instantaneous to the customer as follows. The auto-dialer invites the customer. When the customer picks up, the autodialer invites the telemarketer with "Also: C". The telemarketer then invites the customer with the "Replaces: A". The customer will accept the call (as it is just adding a new party to an existing call), and then immediately drop the autodialer. Through the inclusion of the Requested-By header, the auto-dialer can distinguish the case that the customer is actually talking to the telemarketer, if usually briefly, from being disconnected immediately.

5.3 Multi-Party Conferences

One of the advantages of SIP based telephony is that it enables a wide variety of multi-party conferencing scenarios. These include multicast conferences, bridged conferences, and full-mesh conferences. In a full-mesh conference, each participant sends media data to every other participant and mixes the media from all other participants locally. While network multicast is more efficient for multi-party conferences, a full-mesh may be appropriate for, say, three-party calls or where a bridge or multicast is not available. If DVMRP is used as a multicast routing protocol, small groups are very inefficient, as packets will be periodically broadcast to the whole Internet. Also, it is far easier to eavesdrop on a multicast session. A single SIP conference can combine multicast, full-mesh and a bridge. These services are all possible using only the tools described in Section 4.

5.3.1 Unicast-Based Conferences: Bridges and Meshes

Consider the simplest case, that of a dial-in bridge. In this scenario, users call up a number which represents a bridge. This bridge mixes the media from all users connected to it, and then returns it to each user. In SIP, such a bridge is represented with a SIP URL like any other, e.g., `sip://conf3224@mcus.com`. The caller may not even be aware that this URL is actually a bridge. Each user invites the bridge. The acceptance response describes the media that the bridge can understand, and the port number to send the media to, as with any other call. All users who send an INVITE to the same URL are considered part of the conference. Their media is mixed, and the result is sent to each user in a format they can understand.

RTCP is used to learn about what other parties are in the conference, and to pass around notes (such as far end mute indications) for simple conference functions.

Suppose A who is part of the bridged conference at MCU M would like to call B, not through the bridge, and then invite B to join the bridged conference. To do this, A invites B. After the two connect and talk, A invites B again (same Call-ID), with Also: M. Note that user A's SIP application does not know, or need to know, that M is actually performing a bridge function. In response to the Also, B sends an INVITE to the MCU, with "Requested-By: A". This lets the MCU know that it was A that invited B to join the bridge, and it is thus possible that A is still connected to B directly. To change this, the MCU invites B, with Replaces: A. This causes B to drop A.

If the client does not use Requested-By, the bridge has no way to know which user to place in the Replaces field. To deal with this case, the bridge can use "Replaces: *", which will tell B to disconnect any mesh that existed outside the bridge.

For a full-mesh conference, a participant gets a new participant N to join the mesh by sending it a list of all the other participants it knows about in an Also header (Fig. 5.3.1). (As in [33], we could have each existing party that N calls list the participants it knows about, repairing partially connected meshes.)

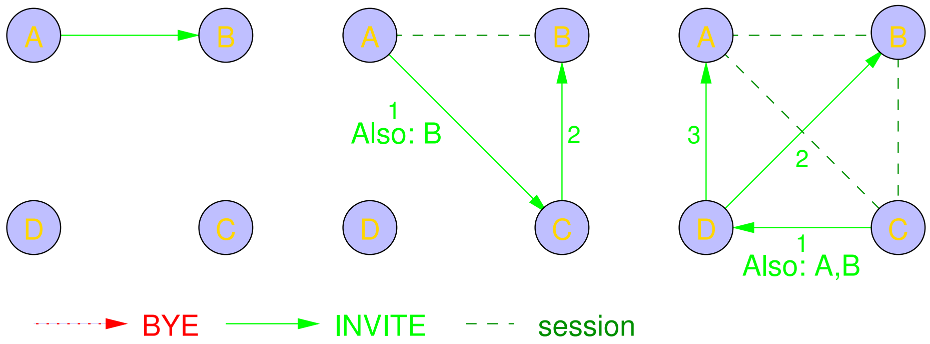

Figure 7: Establishing a full mesh of conference participants

5.3.2 Multicast Conferences

In general, multicast conferences use network resources more efficiently than meshes and bridges. To transition from full mesh to multicast, one user (A) obtains a multicast address, and invites all others in the conferences, with the invitation containing an SDP description which indicates that A wishes to receive its media on the multicast group. Other endpoints which are multicast capable reply with a 200-class response, others that cannot, reply with a 600-class response. This will allow A to know from which users it can expect to receive data through multicast, and which through unicast. Participants which are multicast capable treat the invitation as an opportunity to send a similar invitation themselves. As each participant in the conference sends a multicast invitation to each other, the participants will learn which participants can receive on the multicast group, the media capabilities of each participant in the group. Those participants which cannot receive multicast will continue to receive unicast from each of the other participants. Furthermore, since each participant will know the media capabilities of those receiving from the group, each can send using codecs from the intersection of those capabilities.

If one user (A) wishes to invite a new participant (B) to the conference, the operation is just as if the conference were full mesh. A invites with the Also header listing the other participants in the conference. The SDP in the invitation contains the multicast address, and the media that A is capable of. If B is multicast capable, it replies with an SDP description echoing the multicast address, and indicating its own media capabilities. If B is not multicast capable, it returns an error code to this effect. A can then resend the invitation indicating a unicast address instead. In response to the Also, B will invite the other participants in the conference, indicating unicast or multicast in the SDP, as appropriate.

Even though this conference is multicast, it is still *tightly coupled*, in the sense that when a new participant is invited to the group, it must explicitly invite itself with every other participant. This does not scale well to very large conferences. To deal with this, at any point in time, a participant who wishes to invite a new user may switch to a *loosely coupled* conference mode. To do this, A invites B, as before, but omits the Also list. B will eventually learn about the other group members through RTCP or other media-specific membership announcement mechanism.

The loosely coupled conference model scales better since new members need not know anything about the other members in the conference. However, the new member cannot participate if it is not multicast capable, and it will not be able to communicate with those conference participants who were still connected with a unicast mesh. Note that the decision to switch from tight coupling can be made independently by each participant. There is no need for synchronization.

5.3.3 Switching from a Mesh to a Bridged Conference

To switch from a full mesh to a bridged conference, some user in the conference (A), locates a bridge B, using a mechanism outside the scope of SIP, and invites the bridge, with Also enumerating the other conference participants, similar to a call transfer (Fig. 5.3.3). As the bridge works through the invitation list, it includes all the successfully bridged members in the Replaces header to the new invitees. The SDP description in each of the invitations indicates the capabilities and receive ports of the bridge. Any participant may then invite new members to the conference in the same fashion as for the dial-in bridge scenario described above.

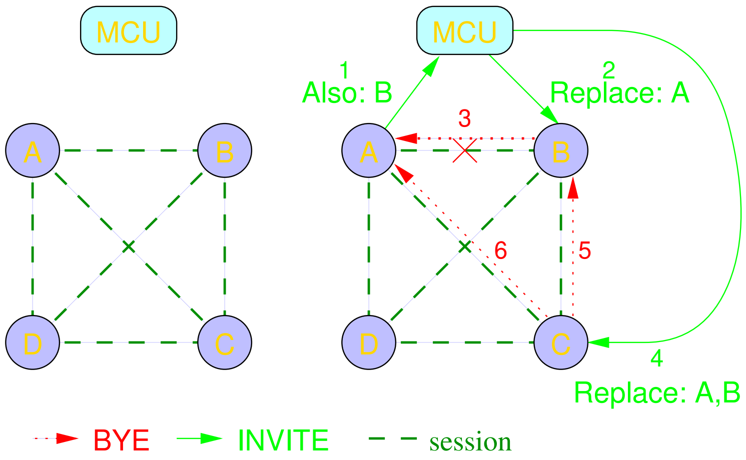

Figure 8: Transitioning from a full-mesh conference to a bridged conference

Note that except for the initiator and bridge, none of the conference participants need know how to interact with a bridge. The participants will all transition to a bridge as a natural consequence of the behavior defined by the Also and Replaces fields. There also is no disruption of the conference. The result of the transition may well be a mixture of a bridged and meshed conference.

5.3.4 Multicast Signaling

SIP also allows a client to send requests via multicast. We can envision a number of different applications for multicast invitations to a conference:

- The client wishes to invite a group of friends, represented by a single identifier (`friends@isp.com`, for example) which maps to a multicast address. Each member of this group listens to the address, and can therefore receive invitations. This is a *wide area reach-all* application.
- A client wishes to invite all members of the department to a conference, without having to keep track of the department members. It would be efficient to send an invitation to a local multicast group representing the department. This is an example of a *local area reach-all* application.
- A researcher wishes to speak to speak to a system administrator, but does not know which one is available. Instead of calling each sequentially, the client can multicast an invitation, and a single administrator responds. This is a version of any-cast, which we refer to as the *local area reach-first* application.
- A client wishes to speak to a member of the board of directors of a club, but does not care which one. The client multicasts an invitation, and a single director responds. This is an example of a *wide area reach-first* application.
- A client wishes to advertise a multicast session, inviting any users who are listening to join the session. The client does not want responses from the users it invites. This is an example of a *wide area advertise* application.

Mapping a SIP URL to a multicast group is beyond the scope of this work; it may involve static configuration for local-area applications or directory services.

In order to scale the reach first and reach all applications for both local and wide area usage, we define some basic rules and protocol mechanisms for multicast usage. Since many parties may send SIP messages to a multicast group (particularly in the advertise application), clients must implement a back-off algorithm before sending a message. This algorithm is called *reconsideration* [10, 34]. It allows for fair distribution of bandwidth among senders in a multicast group with a minimal amount of state storage or complexity. Before a client sends, it listens to the group to see if anyone else is sending. If so, it reschedules its transmission of the request based on the number of other senders heard. In the case of the advertising application, it is desirable to send the message periodically. Reconsideration defines the mechanism by which each client determines the period of the message transmission.

Once the message is multicast, there must be an indication of whether it is to reach all, reach first, or advertise. This is indicated in the Call-Disposition header.

In the case of *advertise*, the servers which receive the request do not respond. Since the request is an INVITE message with a multicast group listed for the conference, the servers can join the group immediately and participate in the session. In the case of *reach first*, things are more complicated. We desire just one response among all receivers that get the invitation. We observe that this is the same as the "NAK-suppression problem" in reliable multicast [35]. Therefore, when a server receives a multicast invitation with a reach-first disposition, it waits a random amount of time T, and then multicasts a response. Should the receiver hear another server respond before it sends its own response, it cancels its own response. If multiple servers send responses back to the group, the one with the lowest unicast address is considered the "winner". To acknowledge this, the client sends its ACK message to the multicast address (to make sure everyone has a consistent view of who the winner was), listing the designated winner in the To field. The winner responds to this message with a unicast response, not multicast.

In networks with lots of servers listening to a multicast address, and where network delays are large, additional means are needed to reduce the response flood. To do this, we mandate that the random variable T is *not* distributed uniformly (as is done in protocols like IGMP [36] and SRM [35]), but rather has a skewed distribution. The skew makes it more likely that a response is sent later rather than earlier. This allows for responses to be sent only sporadically in the beginning, with the response rate picking up as time goes on. However, since a response suppression algorithm is being used, this behavior is desirable - the bulk of users towards the end of the distribution will never send since their response will be suppressed.

In the case of reach-all, we apply the same *reconsideration* algorithm to the responses. Since the client wishes to receive all of the responses, a suppression algorithm is not appropriate. Reconsideration will allow each server to send a response, but will spread them out uniformly over time. Furthermore, a variation of reconsideration, called *unconditional reconsideration*, is particularly effective at preventing floods of packets when many users simultaneously send a response [34], as they are likely to try to do here. The ACK and the responses which result are also sent in the same fashion. This will allow a client to establish a signaling relationship with all of the members of a multicast group in a scalable fashion.

Using multicast invitations is a logical next step in growing a conference beyond the multicast-media conferences discussed above. As the number of members of such a conference becomes very large, a member can multicast an invitation with Call-Disposition set to advertise. This will allow potentially thousands of members to join the group. In this way, SIP provides a flexible framework for signaling for both small conferences and very large conferences alike, allowing for a migration from one to the other. Thus, there is no need for a separate announcement protocol such as SAP [11], although the style of user interaction may well be different. For an advertisement, for example, it is likely that these would only be listed in some temporal or subject ordering, rather than alerting the recipient.

5.4 Mute and Hold Services

The mechanisms described in Section 4 support a wide range of mute and hold services. Near end mute, where the client continues to receive media but does not generate it, requires no protocol assistance. To implement far end mute, a participant need only send an INVITE to another participant, indicating a null set of receive capabilities for any of the media. This will cause the other participant to cease sending that particular media.

Putting another user on hold is trivially supported by ceasing to send media. RTCP can be used to send a note, such as "holding" to assure to user that they are still connected, but put on hold. More interesting services are also possible. Instead of simply ceasing transmission, a user can send another INVITE indicating a multicast address. This multicast address could be fed by a media server which is streaming background music. Since SDP can contain pointers to RTSP [29] content, it is even possible to give users remote control over the music they hear while waiting.

6 Interaction with Stored Media

Since Internet telephony shares protocols and infrastructure with other Internet multimedia services, it is of interest to explore new combinations of services that span several of these. In this section, we look at several examples. While in no way required for use with SIP, the Real-Time Streaming Protocol (RTSP) [29] is a

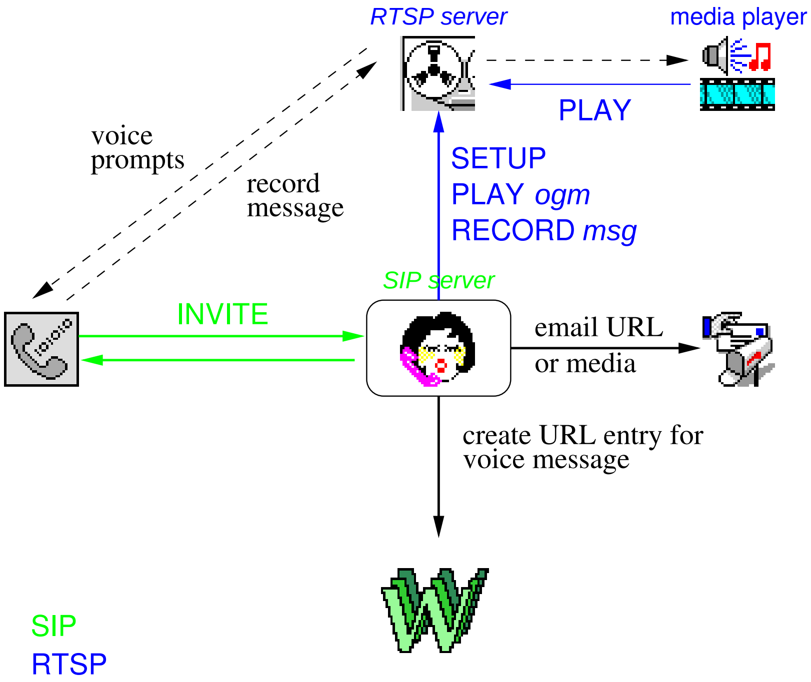

Figure 9: Implementation of voice mail using SIP and RTSP

natural complement to SIP. RTSP's role is to control the delivery of stored multimedia content, including precise control of playback and recording suitable for remote digital linear editing.

Perhaps the best example of this is voice mail. Instead of treating voice mail as a special-case service for telephony only, we can treat it as nothing but a recording and playback service controlled by the IPtel end system or server. There are a number of ways in which the voice-mail service can be accessed. In the most direct approach, a SIP server sends a redirect response back to the client. The Location header contains an RTSP URL. The client then communicates with the RTSP server, recording a message. Here, the client has to be able to understand RTSP.

As an alternative approach, shown in Fig. 6, the SIP server can return a SIP URL which points to a specialized voice-mail server. This server will answer SIP invitations, and establish an actual SIP session with the caller. The voice-mail server can then send RTSP commands to the RTSP server to play the outgoing message, and record whatever media is received from the caller.

After completing the recording, there are several methods by which the recording can be made available to the recipient:

- The server can mail the multimedia file as a MIME attachment to the recipient, so that it appears integrated with the remainder of the user's email and can be filed, forwarded and replied to. (Again, the use of email addresses for SIP helps here.)
- The server can email an RTSP URL to the recipient, pointing to the recording, avoiding the transmission of possibly large multimedia files.
- The server can email an HTTP URL to the recipient. This HTTP URL actually points to a Java applet. This applet contains a user interface and RTSP implementation that contacts the media server. This is similar to the previous approach, but avoids the need for the user to even have an RTSP implementation in their mail reader.

- The server can transfer the message to a traditional telephony voice-mail server. (The method of such a transfer is outside the scope of this paper.) The user can access voice-mail messages in an integrated fashion.

RTSP can also play a role in conferences as a device for recording and playback of conferences. A conference participant can invite a SIP-speaking RTSP server, bringing it into an existing conference, appearing as just another participant in the conference. Alternatively, for multicast conferences, an RTSP server can simply be given the same session description as was used for invitations.

7 Related Work

Efforts to design multimedia applications and protocols for packet-switched networks [37, 38] date back to the early days of the Internet; systems have been developed for various combinations of packet-switched and circuit-switched networks [39, 40]. In particular, the set of tools commonly known as the Mbone conferencing tools [9, 41, 42, 43] has achieved widespread use. Examples of multimedia control include Etherphone [44], Rapport [45] and MMCC [46, 47]. Etherphone and MMCC are based on a centralized control model, while SIP has no notion of a conference controller or similar device. MMCC did not incorporate call control functionality. The H.323 protocol suite [1], based on the ISDN Q.931 protocols, is being defined as a framework for Internet telephony call control services. SIP attempts to offer a generalized set of services which is not encumbered by H.323's ISDN legacy. SIP is based on work reported in [48] and [49].

8 Conclusion and Future Work

We have described a protocol for Internet telephony signaling, the Session Initiation Protocol. SIP provides a framework for complex and rich telephony services, including user location, forward, transfer, multiparty, mute and hold. It is simple, flexible, extensible, and based on existing architectures, such as HTTP and DNS. We have described how to instantiate specific services with the primitives provided by SIP. Space does not permit to describe other services that SIP can support, including automatic call distribution and interactive voice response systems, as well as possibly a light-weight form of terminal mobility. The more open service creation environment requires cryptographic security rather than relying on the physical separation and trusted providers in the PSTN; work is under way to extend the S/MIME and PGP email security mechanisms to SIP to support authenticated and/or private signaling. Our work continues on defining additional services, and extending the multiparty scenarios into more advanced cases.

SIP is currently being standardized within the Internet Engineering Task Force (IETF) [50].

References

[1] 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.

[2] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 1771, Internet Engineering Task Force, Mar. 1995.

[3] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP) – version 1 functional specification," RFC 2205, Internet Engineering Task Force, Oct. 1997.

[4] P. Pan and H. Schulzrinne, "Yessir: A simple reservation mechanism for the internet," Technical Report RC 20697, IBM Research, Hawthorne, New York, Sept. 1997.

[5] C. Rigney, "RADIUS accounting," RFC 2139, Internet Engineering Task Force, Apr. 1997.

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

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

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

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

[10] J. Rosenberg and H. Schulzrinne, "Timer reconsideration for enhanced RTP scalability," Internet Draft, Internet Engineering Task Force, July 1997. Work in progress.

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

[12] M. Handley, "SDP: Session description protocol," Internet Draft, Internet Engineering Task Force, Nov. 1997. Work in progress.

[13] L. Masinter, P. Hoffman, and J. Zawinski, "The mailto URL scheme," Internet Draft, Internet Engineering Task Force, Oct. 1997. Work in progress.

[14] T. Howes, S. Kille, and M. Wahl, "Lightweight directory access protocol (v3)," RFC 2251, Internet Engineering Task Force, Dec. 1997.

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

[16] C. Rigney, "RADIUS accounting," RFC 2059, Internet Engineering Task Force, Jan. 1997.

[17] F. Yergeau, "UTF-8, a transformation format of unicode and ISO 10646," RFC 2044, Internet Engineering Task Force, Oct. 1996.

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

[19] J. Franks, P. Hallam-Baker, and J. Hostetler, "An extension to HTTP: digest access authentication," RFC 2069, Internet Engineering Task Force, Jan. 1997.

[20] 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.

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

[22] D. Kristol and L. Montulli, "HTTP state management mechanism," RFC 2109, Internet Engineering Task Force, Feb. 1997.

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

[24] C. Allen and T. Dierks, "The TLS protocol version 1.0," Internet Draft, Internet Engineering Task Force, Oct. 1997. Work in progress.

[25] B. Hinden and C. Partridge, "Version 2 of the reliable data protocol (RDP)," RFC 1151, Internet Engineering Task Force, Apr. 1990.

[26] C. Allen and T. Dierks, "The TLS protocol version 1.0," Internet Draft, Internet Engineering Task Force, Nov. 1997. Work in progress.

[27] International Telecommunication Union, "Control protocol for multimedia communication," Recommendation H.245, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mar. 1996.

[28] D. Zimmerman, “The finger user information protocol,” RFC 1288, Internet Engineering Task Force, Dec. 1991.

[29] H. Schulzrinne, A. Rao, and R. Lanphier, “Real time streaming protocol (RTSP),” Internet Draft, Internet Engineering Task Force, Oct. 1997. Work in progress.

[30] R. Williams, “User location service,” Internet Draft, Internet Engineering Task Force, Feb. 1996. Work in progress.

[31] W. W. Gaver, “Sound support for collaboration,” in *Proceedings of the Second European Conference on Computer-Supported Cooperative Work (ECSCW’91)* (L. Bannon, M. Robinson, and K. Schmidt, eds.), (Amsterdam, The Netherlands), pp. 293–308, Amsterdam, Sept. 1991.

[32] J. C. Tang and M. Rua, “Montage: Providing teleproximity for distributed groups,” in *Proc. of*, (Proceedings of the Conference on Computer Human Interaction (CHI) ‘94), pp. 37–43, Apr. 1994.

[33] C. Elliott, “A ’sticky’ conference control protocol,” *Internetworking: Research and Experience*, vol. 5, pp. 97–119, 1994.

[34] J. Rosenberg and H. Schulzrinne, “Timer reconsideration for enhanced RTP scalability,” in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), March/April 1998.

[35] 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*, (Cambridge, Massachusetts), pp. –, Sept. 1995.

[36] B. Fenner, “Internet group management protocol, version 2,” Internet Draft, Internet Engineering Task Force, Oct. 1997. Work in progress.

[37] 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.

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

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

[40] 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.

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

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

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] H. Schulzrinne, “Personal mobility for multimedia services in the Internet,” in *European Workshop on Interactive Distributed Multimedia Systems and Services*, (Berlin, Germany), Mar. 1996.

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

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