Signaling for Internet Telephony

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

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

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.

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. A call may encompass a number of *connections*, where a connection is a logical relationship between a pair of end systems in a call. For example, a non-bridged three party audio only call will have three connections, creating a full mesh among the participants. A *media stream or session* is the flow of a single type of media among a set of users. This flow can either be unicast (in which case it is between two users), or multicast. A media session is associated with one or more connections. In the above three party call example, if the media is distributed using unicast, there will be one audio session per connection. If the audio is distributed via multicast, there will be one audio session associated with all three connections. We do not require that calls have media streams associated with them, but this is likely to be the common case.

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 3.2.

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 connections may involve different media and media parameters. Many software codecs are able to receive different encodings while being restricted to sending one type of media for each stream.

Any call participant can invite others into an existing call (thus establishing connections) and terminate connections with others; we call this function (*call participant management*). Call participant management also encompasses transfer, hold, and transitions among unicast, multicast, and bridged media distribution.

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 [30] 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.

In Section 2 we motivate the need for SIP, and discuss the basics of SIP operation, its addressing structure, message syntax and transport. In Section 3, we discuss how SIP can be used for telephony services, focusing on how services are constructed from simple primitive tools. We then mention related work in Section 4, and conclude in Section 5.

2. The Session Initiation Protocol

2.1. Motivation

The study of protocols for conference control is certainly not new. In fact, numerous protocols have even been standardized for conference services in the Internet, most prominent among them the H.323 suite from the ITU-T. However, we find that H.323 suffers from a number of serious drawbacks [28]:

Complexity The H.323 specifications are over 700 pages and growing; implementations in C++ require near 100,000 lines of code.

Scalability H.323 was engineered for conferences on a LAN. It relies on a central conference server for almost all functions, and doesn't scale to large conferences (in fact, a separate specification was written to address broadcast environments). Its naming and call routing functions are non-existent, making its use for wide area telephony problematic.

Extensibility As its based on ASN.1 and the Packed Encoding Rules, adding new elements, features, and headers, and managing compatibility across different versions, is not easily done. H.323 also only works with a small number of standardized ITU speech and video codecs.

It was therefore our goal with SIP to leverage off the work on distributed conferencing services [6, 21, 25] and existing Internet protocols (namely the Hypertext Transfer Protocol (HTTP) [8] and the Simple Mail Transfer Protocol (SMTP) [18, 5, 4] to create a powerful, flexible, simple, and scalable protocol that could serve as a real foundation for true wide area Internet telephony.

2.2. Role of SIP

As mentioned previously, the role of SIP is to seek out, locate, and invite participants to an IP telephony call. 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. Note also that a participant can be a human user, an automaton (such as a media server), or a gateway to some other network, such the the PSTN or an H.323 network.

SIP performs its function of "seeking out" a user by defining mechanisms for SIP call invitations to traverse servers throughout the network. A server either knows the location for a user, or it forwards the invitation to a server which it believes knows the location of a user. This allows for mobility services; the type and range of which is defined by program logic on each server.

SIP is also responsible for basic call management. This includes the ability to initiate a call, terminate a call, and to add and remove users from a call. SIP is also responsible for setting and changing the media content of the session. It accomplishes this by delivering opaque media session descriptions.

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 conference with thousands of members.

SIP can be used to initiate multicast or unicast media sessions. SIP messages themselves can be either unicast or multicast. Not all combinations make sense, of course. For example, inviting somebody to a multicast conference (e.g., of the type found on the Mbone [7]) requires unicast signaling. For automatic call distribution (ACD), where a caller wants to reach the first available person, multicast signaling may be useful, yet the conference may be unicast. Finally, one may want to invite groups of people to a multicast conference, in which case multicast signaling is appropriate. 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 [20]. Multicast signaling also requires that potential invitees already expect an invitation, i.e., subscribe to the multicast group.

2.3. Overview

SIP is a client-server protocol, with requests issued by the client and responses returned by the server. In the case of IP telephony, the caller acts as client, and the callee acts as a 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. A single host may well act as client and server for the same call, as discussed in Section 2.6.

As in HTTP, the client requests invoke *methods* on the server. Requests and responses are textual 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 and mail servers.

Calls in SIP are uniquely identified by the a call identifier, carried in the Call-ID header field in SIP messages. The call identifier is created by the creator of the call and used by all call participants. Connections have the following properties: The *logical connection source* indicates the entity that is requesting the connection (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 connection 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 connection destination. *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) [11] serves this purpose, 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.

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. CANCEL terminates a search for a user. Finally, REGISTER conveys location information to a SIP server.

2.4. 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 (Public Switched Telephone Network (PSTN)) addresses, OSI addresses or private numbering plans.

2.5. 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 address 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` [17] URL today.

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 [13], 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 their 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.

2.6. 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 [10], 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 1 and 2 show the behavior of SIP proxy and redirect servers, respectively.

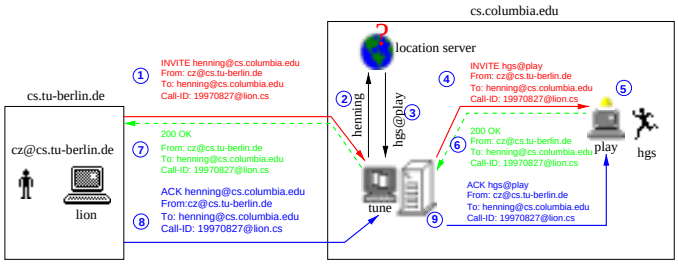

Figure 1. SIP invitation in proxy mode

Once the user agent server has been contacted, it sends a response back to the client. The response has a response code and reason phrase. 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. To ensure reliability even with unreliable transport protocols, the server retransmits final responses until the client confirms receipt by sending an ACK request to

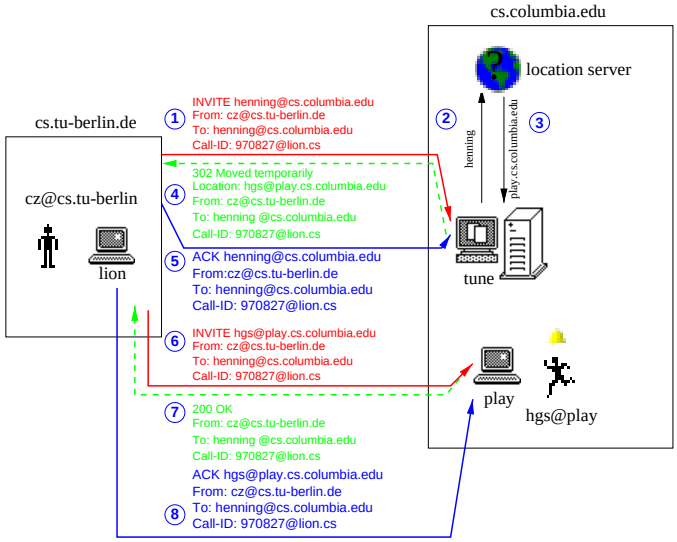

Figure 2. SIP invitation in redirect mode

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).

2.7. Proxy Servers

A proxy server (or a redirect server, for that matter), has as its main function the forwarding or redirection of requests to some other server. To determine which server should be contacted for a particular call, the server invokes a *Locator Service*. The locator service is a completely local operation, and may be the result of a database lookup, finger operation, or result of some programmatic operation on the incoming request.

SIP itself provides a simple locator service by means of the REGISTER message. A SIP client sends this message to its server, indicating its physical location and a name to associate with that location. In this fashion, a SIP server can build up a database of translations for those users in its domain. SIP REGISTER messages can also be multicast, so that a client which doesn't know its server can discover the nearest one and register with it.

It is possible that the locator service may yield multiple potential locations for a particular user. Proxy servers can forward an 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.

When an a call invitation is forwarded by a proxy to multiple servers at once, it becomes possible for the invitation to generate multiple call acceptances. SIP allows for this condition, so that a proxy server can return several positive responses to the client. However, a proxy server can only ever return a single negative response, and only if it never returned a positive one. As a result, a caller will know definitively that the call is not accepted if a negative response comes back. However, if it receives an accept, the caller can choose to wait for further acceptances if it desires, or just speak with the first. There are many possible actions a caller can take upon receiving multiple acceptances. They can reject all but one of the parties, choosing the most appropriate one (for example, if a human user and an answering machine pick up, the user can choose to speak to the human. SIP conveys information about the endpoint when it answers a call, so a caller can easily make the distinction), they can talk to all of them, forming a multiparty conference, or they can place some of them on hold. The possibilities are limited only by the software in the calling system.

A SIP proxy server also has the option of being stateless or stateful. A stateless server receives a request, forwards to the next hop server, and forgets. The Via field in the request and response carry the only piece of state necessary for operation: the previous hop of the request. Therefore, when a stateless proxy receives a response, it simply looks at the Via field and extracts the previous hop to forward the response to. SIP also allows for a server to transition from stateful to stateless mid-transaction. This allows for failure modes and for operation with fixed amounts of memory, improving scalability. Larger SIP servers can operate in stateless mode, while smaller, stateful servers at domain peripheries can operate in a stateful manner.

3. 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.

These requirements, in fact, are much like those which pushed the Intelligent Networking standards. However, unlike IN, which focuses on allowing service providers to rapidly deploy services inside the network, our goal is to expose these kinds of features to end systems.

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.

3.1. 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 service. 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 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. 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.

3.2. 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 [31], 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 some other 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 3, 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 [27] 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.

3.3. 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 to 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 sends a BYE to 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: C", 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.

3.4. 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. A single SIP conference can combine multicast, full-mesh and a bridge. These services are all possible using only the tools described in Section 3.

3.5. 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.

The Real Time Control Protocol (RTCP) [26] 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. 3). (As in [6], we could have each existing party that N calls list the participants it knows about, repairing partially connected meshes.)

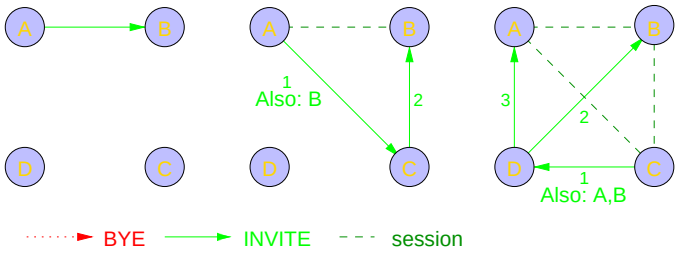

Figure 3. Establishing a full mesh of conference participants

3.6. 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 sends an INVITE to all others in the conference. The invitation contains 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. 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.

3.7. 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. 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.

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.

3.8. Mute and Hold Services

The mechanisms described in Section 3 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 [27] content, it is even possible to give users remote control over the music they hear while waiting.

4. Related Work

Efforts to design multimedia applications and protocols for packet-switched networks [16, 2] date back to the early days of the Internet; systems have been developed for various combinations of packet-switched and circuit-switched networks [3, 22]. In particular, the set of tools commonly known as the Mbone conferencing tools [7, 15, 9, 24] has achieved widespread use. Examples of multimedia control include Etherphone [19], Rapport [1] and MMCC [23, 21]. 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 [14], 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 [25], [12], and [29].

5. 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. 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) [12].

References

[1] S. R. Ahuja and J. R. Ensor. Call and connection management: making desktop conferencing systems a real service. *ACM Computer Communication Review*, 22(3):10–11, Mar. 1992.

[2] Anonymous. Special issue on packet switched voice and data communication. *IEEE Journal on Selected Areas in Communications*, SAC-1(6), Dec. 1983.

[3] Arango et al. Touring machine system. *Communications ACM*, 36(1):68–77, Jan. 1993.

[4] N. Borenstein and N. Freed. MIME (multipurpose internet mail extensions): Mechanisms for specifying and describing the format of internet message bodies. RFC 1341, Internet Engineering Task Force, June 1992.

[5] D. Crocker. Standard for the format of ARPA internet text messages. RFC STD 11, 822, Internet Engineering Task Force, Aug. 1982.

[6] C. Elliott. A 'sticky' conference control protocol. *Internetworking: Research and Experience*, 5:97–119, 1994.

[7] H. Eriksson. MBONE: The multicast backbone. *Communications ACM*, 37(8):54–60, Aug. 1994.

[8] R. Fielding, J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2068, Internet Engineering Task Force, Jan. 1997.

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

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

[11] M. Handley and V. Jacobson. SDP: session description protocol. RFC 2327, Internet Engineering Task Force, Apr. 1998.

[12] M. Handley, H. Schulzrinne, and E. Schooler. SIP: session initiation protocol. Internet Draft, Internet Engineering Task Force, May 1998. Work in progress.

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

[14] 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.

[15] V. Jacobson. Multimedia conferencing on the Internet. In *SIGCOMM Symposium on Communications Architectures and Protocols*, London, England, Aug. 1994. Tutorial slides.

[16] D. T. Magill. Adaptive speech compression for packet communication systems. In *Conference record of the IEEE National Telecommunications Conference*, pages 29D–1 – 29D–5, 1973.

[17] L. Masinter, P. Hoffman, and J. Zawinski. The mailto URL scheme. Internet Draft, Internet Engineering Task Force, Jan. 1998. Work in progress.

[18] J. Postel. Simple mail transfer protocol. RFC STD 10, 821, Internet Engineering Task Force, Aug. 1982.

[19] 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*, 9(9):1395–1404, Dec. 1991.

[20] 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.

[21] E. Schooler and S. L. Casner. An architecture for multimedia connection management. *ACM Computer Communication Review*, 22(3):73–74, Mar. 1992.

[22] E. M. Schooler and S. L. Casner. A packet-switched multimedia conferencing system. *SIGOIS (ACM Special Interest Group on Office Information Systems) Bulletin*, 10(1):12–22, Jan. 1989.

[23] 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*, volume 3, pages 707–716, Hawaii, Jan. 1991. IEEE.

[24] 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.

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

[26] 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.

[27] H. Schulzrinne, R. Lanphier, and A. Rao. Real time streaming protocol (RTSP). RFC 2326, Internet Engineering Task Force, Apr. 1998.

[28] H. Schulzrinne and J. Rosenberg. A comparison of SIP and H.323 for internet telephony. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, England, July 1998.

[29] H. Schulzrinne and J. Rosenberg. SIP call control services. Internet Draft, Internet Engineering Task Force, Feb. 1998. Work in progress.

[30] G. A. Thom. H.323: the multimedia communications standard for local area networks. *IEEE Communications Magazine*, 34(12):–, Dec. 1996.

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