♦ The Session Initiation Protocol: Providing Advanced Telephony Services Across the Internet

Henning G. Schulzrinne and Jonathan D. Rosenberg

During the past few years, Internet telephony has evolved from a toy for the technically savvy to a technology that, in the not too distant future, may replace the existing circuit-switched telephone network. Supporting the widespread use of Internet telephony requires a host of standardized protocols to ensure quality of service (QoS), transport audio and video data, provide directory services, and enable signaling. Signaling protocols are of particular interest because they are the basis for advanced services such as mobility, universal numbers, multiparty conferencing, voice mail, and automatic call distribution. Two signaling protocols have emerged to fill this need: the ITU-T H.323 suite of protocols and session initiation protocol (SIP), developed by the Internet Engineering Task Force (IETF). In this paper we examine how SIP is used in Internet telephony. We present an overview of the protocol and its architecture, and describe how it can be used to provide a number of advanced services. Our discussion of some of SIP's strengths—its simplicity, scalability, extensibility, and modularity—also analyzes why these are critical components for an IP telephony signaling protocol. SIP will prove to be a valuable tool, not just for end-to-end IP services, but also for controlling existing phone services.

Introduction

The term *Internet telephony* has evolved to infer a range of different services. In general, it refers to the transport of real-time media—such as voice and video—over the Internet to provide interactive communications among Internet users. The parties involved may access the Internet via a PC, a stand-alone Internet protocol (IP)-enabled device, or even by dialing up to a gateway from the handset of a traditional public switched telephone network (PSTN).

The advantages of IP telephony are quite sweeping.[1] It offers high-quality voice, improved multiplexing gains (an inherent advantage in any packet-switched network), rich computer telephony integration, advanced services, an open market for providers, and reduced cost.

Supporting Internet telephony requires a variety of components, each of which can be supported by several protocols:

- Transport, provided by the real-time transport protocol (RTP);[2]
- QoS, provided by the resource reservation protocol (RSVP),[3,4] yet another sender session Internet reservations (YESSIR)—both of which make end-to-end resource reservations on the Internet[5]—or differentiated services;[6]
- Authentication, authorization, and accounting, provided by the remote authentication dial-in user service (RADIUS)[7,8] and DIAMETER;[9]
- Gateway discovery,[10] provided by the gateway location protocol;[11] and

Panel 1. Abbreviations, Acronyms, and Terms

ACD—automatic call distribution
BGP—border gateway protocol
CGI—common gateway interface
DIAMETER—a protocol for authentication, authorization, and accounting
DNS—Domain Name System
GSTN—global switched telephone network
HTML—HyperText Markup Language
HTTP—hypertext transport protocol
IANA—Internet Assigned Numbers Authority
IETF—Internet Engineering Task Force
IN—intelligent network
IP—Internet protocol
IPtel—Internet protocol telephony
ISDN—integrated services digital network
ISP—Internet service provider
ITU–T—International Telecommunication Union–Telecommunication Standardization Sector
JTAPI—Java telephony application programming interface
LDAP—lightweight directory access protocol
Mbone—multicast backbone
MIME—multipurpose Internet mail extension
MTA—message transfer agent
MX—mail exchange
PBX—private branch exchange
PEP—protocol extensions protocol
Perl—Practical Extraction Report Language, a text processing language
PINT—PSTN and IP Internetworking
POTS—"plain old telephone service"
PSTN—public switched telephone network
QoS—quality of service
RADIUS—remote authentication dial-in user service
RSVP—resource reservation protocol
RTP—real-time transport protocol
RTSP—real-time streaming protocol
SCP—service control point
SDP—session description protocol
sdr—an mbone session directory tool
SIP—session initiation protocol
SMIL—Synchronized Multimedia Integration Language
SMTP—simple mail transfer protocol
SRV—a DNS resource record for servers
TCP—transmission control protocol
UAC—user agent client
UAS—user agent server
UDP—user datagram protocol
URI—uniform resource identifier
URL—uniform resource location
VPN—virtual private network
XML—Extensible Markup Language
YESSIR—yet another sender session Internet reservations

- Directory service, provided by the lightweight directory access protocol (LDAP).[12]

In addition, Internet telephony requires a means for prospective communications partners to find each other and to signal to one another their desire to communicate. We refer to this functionality as *Internet telephony signaling*.

Internet telephony signaling encompasses a number of functions, including:

- *Name translation and user location*, which involve the mapping between names to identify a callee and the eventual location of the callee, be it a telephone number, pager, voice mail, e-mail address, or Web page. This translation and location function can be more complex than a simple database lookup, but it may depend on caller and callee preferences; media and codec support at various points of presence; and service provider, or third-party, logic.
- *Feature negotiation*, which allows a group of participants to agree on the media to exchange, and their respective parameters. In a multiparty IP telephony conference, the set and type of media need not be uniform. Different participants can exchange different media types with each other; some may only receive audio, while others may receive audio and video. Such a mix may be due to limited computing facilities at an endpoint, a desire to use a particular format for a medium, or the lack of a single medium format common to all participants.
- *Call participant management*, which allows any call participant to invite new users into an existing call and terminate associations with other participants. During the call, participants

should be able to transfer other participants and place them on hold.

- *Call feature changes*, which 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 adding or removing call participants.

Two protocols have emerged to provide these functions, the International Telecommunication Union—Telecommunication Standardization Sector (ITU-T) H.323 series of recommendations,[13] and the Internet Engineering Task Force (IETF) session initiation protocol (SIP).[14] This paper presents SIP and describes how it is used for various advanced services. In an earlier work we compared these services with H.323.[15] The remainder of this paper is organized as follows: First, we present an overview of SIP and its basic functionality, followed by an analysis of several network services enabled by SIP. Next we describe how SIP supports client-to-client services, and then we review its benefits.

Overview of SIP

SIP—used to establish, change, and tear down calls between one or more endpoints in an IP-based network—is based heavily on some of the most successful protocols to emerge from the IETF, which standardizes protocols used on the Internet. In particular, SIP is modeled after the simple mail transfer protocol (SMTP),[16,17] the basis for e-mail, and the hypertext transfer protocol (HTTP),[18] the basis of the Web. Like both of these, SIP is a textual client-server protocol, in which requests are issued by the client and responses are returned by the server.

SIP reuses much of the syntax and semantics of HTTP, including its response code architecture, many message headers, and its overall operation. SIP maps each IP telephony function to one or more transaction requests issued by a client, and one or more responses returned by one or more servers. Like HTTP transactions, SIP transactions are idempotent. Also like HTTP, each SIP request is an attempt to invoke some method on the server. Of the six SIP methods that exist, the most basic is the `INVITE` method, used to initiate a call between the client and the server.

Unlike HTTP and SMTP, SIP can run on top of either the transmission control protocol (TCP) or the user datagram protocol (UDP). SIP provides its own mechanisms for reliability, and UDP enables SIP messages to be multicast. Multicasting allows for, among other features, group invitations and basic automatic call distribution (ACD) functions that do not require a distribution server. By avoiding the TCP synchronization handshake, UDP facilitates fast operation; removing the need for the TCP state in the kernel provides better scalability. When used with TCP, SIP allows many requests and responses to be sent over the same TCP connection, as in HTTP 1.1.[18]

Protocol Components

A SIP system has only two components: user agents and network servers. A *user agent* is an end system that acts on behalf of someone who wants to participate in calls. In general, a user agent contains both a protocol client—called a user agent client (UAC)—and a protocol server—called a user agent server (UAS). The UAC is used to initiate a call, and the UAS is used to answer a call. The presence of both in a user agent enables peer-to-peer operation to take place using a client-server protocol.

In addition to user agents, SIP provides for two different types of network servers: proxy and redirect. A SIP proxy acts in much the same way as an HTTP proxy or an SMTP message transfer agent (MTA). It receives a request, determines which server to send it to, and then forwards the request, possibly after modifying some of the header fields. A SIP proxy has no way of knowing whether the next server to receive the request is another proxy server, a redirect server, or a UAS. For this reason, SIP requests can traverse many servers on their way from UAC to UAS. Responses to a request always travel along the same set of servers the request followed, but in reverse order.

A redirect server receives requests, but instead of forwarding them to the next hop server, it tells the client to contact the next hop server directly. It answers the client's request using a redirect response, which contains the address of the next hop server. This procedure is analogous to iterative searches in the Domain Name System (DNS),[19,20] just as proxying is analogous to recursive searches.

SIP Network Servers

The main function of a SIP network server is to provide for name resolution and user location. When a user wants to place a call, the SIP UAC sends an INVITE request. In general, the caller will not know the IP address or host name of the UAS for the given user; it will only have a name—usually an e-mail address, but sometimes a telephone number or another local identifier—that represents the caller. Using this name, the UAC can determine which network server may be able to resolve the name to an IP address. This network server may, in turn, proxy or redirect the call to additional servers, eventually arriving at one that definitively knows the IP address where the user can be contacted. The process of determining the next-hop server is known as *next-hop routing*. Similar to other dynamic routing protocols, like the border gateway protocol (BGP),[21] SIP provides facilities for loop detection and prevention.

To determine the next-hop server, a SIP network server can use any means at its disposal, such as searching the DNS, accessing databases, executing programs, or prompting users. The final UAS contacted by the caller is determined by the composition of the decisions made at all the servers from caller to callee. The ability of SIP servers to route calls based on any means at their disposal makes them the basis for powerful mobility and forwarding services.

As a result of its next-hop routing decision, a SIP network server may determine that several next-hop servers may be able to contact the user. In these cases, SIP allows a proxy server to *fork* an incoming request, sending it in parallel to multiple next-hop servers. Under normal conditions, each server will generate a response; SIP has rules for merging and returning these responses to the UAC.

Each SIP transaction can take a different path through servers in the network. In a typical call, the first request is an INVITE, which may traverse many network servers on its way to the callee. The response to the INVITE contains a reach address that can be used by the UAC to send subsequent transactions directly to the UAS. Because SIP network servers need not maintain the call state once a transaction is complete, a SIP server has no recollection of the caller or callee. This characteristic facilitates the scalability and reliability of a SIP server, because it can crash and recover (or can swap in a backup), without affecting any of the calls initiated through it. The duration of and amount of state maintained at a server are small compared to those in the global switched telephone network (GSTN), where a switch must maintain the call state for the entire duration of a call. However, a server that wishes to maintain the call state may do so. Through SIP's Route and Record-Route header fields, each proxy individually can insist on being on the signaling path for subsequent transactions. Furthermore, a proxy can change its mind and remove itself from the signaling path later on.

Interestingly, a SIP network server is not required to be *stateful*—that is, to maintain its state—even for the duration of a transaction. A proxy or redirect server can be completely stateless. After it receives a request, it either generates a response or proxies the request, and then forgets everything. The messages themselves contain all the information needed for a stateless proxy to correctly process and route them. This behavior aligns nicely with the Internet datagram architecture, whose packets contain enough information to be individually routed. Furthermore, a stateful proxy can decide to become stateless at any time during a transaction, and the system will still operate correctly. The administrator decides, on a call-by-call basis, whether a proxy will be stateless or stateful. This flexibility of state allows large, central SIP servers to be stateless, but also enables smaller, localized servers to be stateful.

Figure 1 depicts a typical SIP deployment of network servers and the message flow among them. Of the three domains shown (A, B, and C), each has a single SIP server acting as an access point into and out of the networks. Joe, a user agent in domain A, makes a call to Bob, another user. The call invitation is forwarded to the access server of domain A (1), which attempts to find the callee in domains B and C by forking the request.

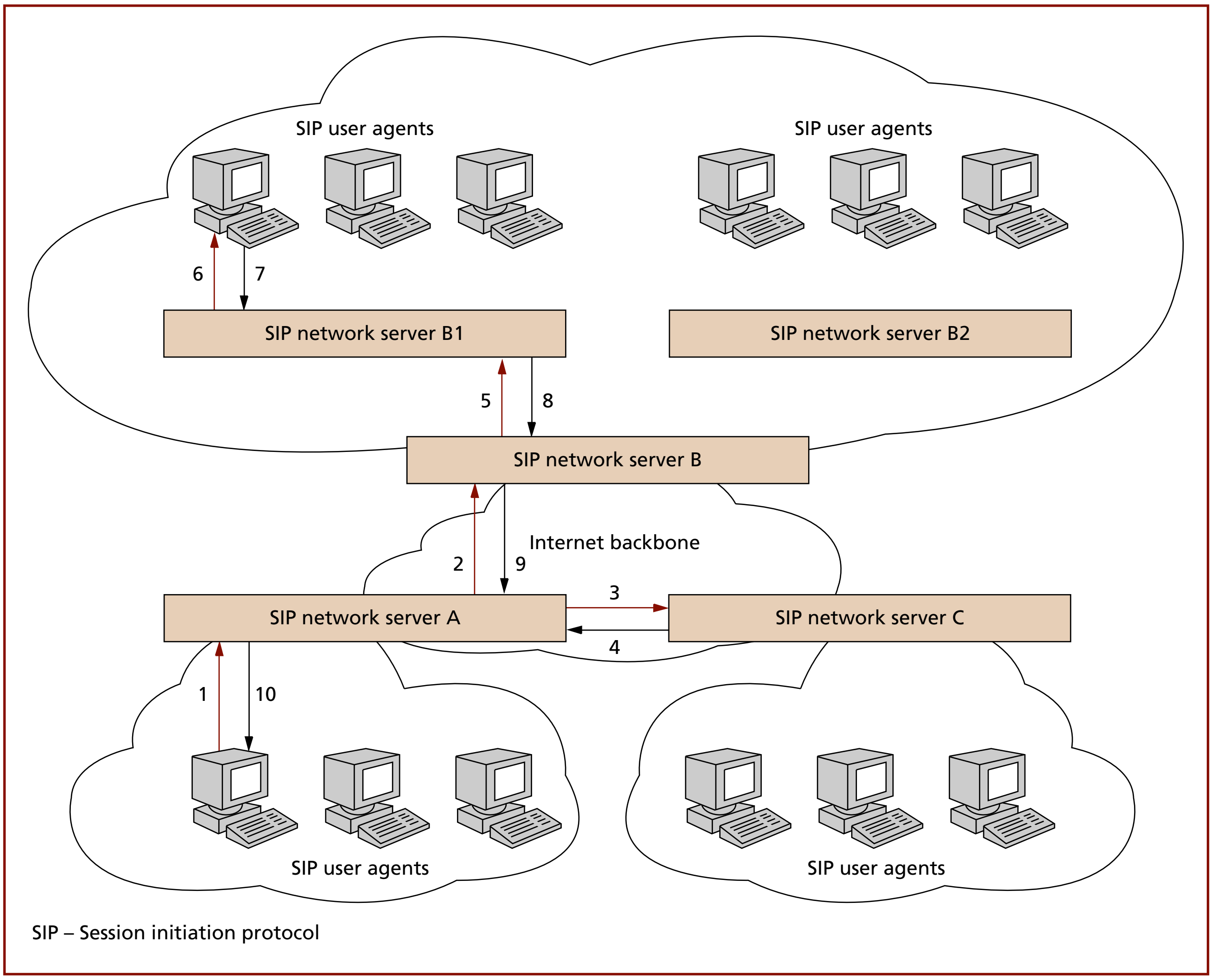


Figure 1.
Typical SIP deployment.

The request arrives at domain C (3), but the user is unknown at this location, so an error response is returned (4). The request at domain B (2), however, is forwarded to a local server internal to domain B (5), where it finally reaches the UAS (6). The response is then returned along the same path to the caller (7,8,9,10).

SIP Messages

A SIP request consists of a request line, header fields, and a message body. The various header fields contain information on call services, addresses, and protocol features. The body, opaque to SIP, can contain anything.

SIP defines several methods, including `INVITE`, `BYE`, `OPTIONS`, `ACK`, `REGISTER`, and `CANCEL`. `INVITE` is used to invite a user to a call. The header fields of an `INVITE` request contain the addresses of the caller and callee, subject of the call, call priority, call routing requests, caller preferences for user location, and desired features of the response, among others. The body of the request contains an opaque description of the media content of the session. Usually, this body is an object described by the session description protocol (SDP),[22] a textual syntax for describing unicast and multicast multimedia sessions. It contains information on codecs, ports, and protocols to be used for sending media to the caller, such as parameters for RTP.[2]

In the case of a broadcast-style multicast-based conference, the body of the request also contains information that describes the content, start and stop times, and administrator of the multicast session. A caller can use this information to invite a callee to participate in an existing multicast backbone (Mbone)[23] session, for example. The amount of information found in SDP is sufficient to allow a caller to begin sending and receiving audio immediately. The response to the INVITE contains the media information for the callee. SDP enables a user to indicate the ability to send and receive with multiple audio and video codecs, and SDP can also rank those codecs in preference of usage. Based on experience, this information provides a basic, but sufficient, means for exchanging media capabilities.

Because the body is opaque to SIP, but described using multipurpose Internet mail extension (MIME)[24] syntax and semantics, SIP can use other media description formats besides SDP. These include H.245[25] capability descriptors, Synchronized Multimedia Integration Language (SMIL)[26] presentation descriptions, and Extensible Markup Language (XML)[27] formatted descriptions of a new video codec. As these and other descriptions appear, they are incorporated into SIP as easily as a new image type is incorporated into HTTP.

REGISTER, which conveys location information to a SIP server, allows a user to tell a SIP server how to map an incoming address into an outgoing address that will reach that user (or another proxy that knows how to reach that user). The body of a REGISTER message can be anything. Currently, researchers are investigating the use of simple scripts to describe the more complex programmatic name translations. This feature, currently planned for standardization within the IETF Internet Protocol Telephony (IPtel) working group, is similar to using service logic for switches. Furthermore, the body of a REGISTER response can contain configuration information useful to the user agent. Such information may include speed dial button configurations; additional addresses that allow for private branch exchange (PBX)-like functionality, whereby the server chooses the addresses used by each client; or a call log. Using the multipart MIME formatting rules and the capability negotiation features of HTTP, new types of information can be added as time passes.

Beyond INVITE and REGISTER, described in the section above, the SIP methods also include:

- BYE, which terminates a connection between two users in a conference;
- OPTIONS, which solicits information about the capabilities of the callee, but does not set up a call;
- ACK, which confirms reliable message exchanges; and
- CANCEL, which terminates a pending request, but does not undo a completed call. When received at a UAS, CANCEL has no effect if the UAS has already answered the call. If the UAS has not answered, CANCEL indicates that it should not bother responding because the call has effectively been canceled. This does not, however, prevent a UAS from answering the call request. It is simply an optimization.

Because SIP is a textual protocol, generation and parsing of its messages are done trivially, particularly with powerful text processing languages such as the Practical Extraction Report Language (Perl). Furthermore, its compliance to RFC 822[28] formatting rules means existing HTTP or SMTP parsers can be used directly for lexicographic analysis of messages. It also vastly simplifies debugging, development cycles, and extensions. **Figure 2** shows a typical SIP INVITE.

Addressing and Naming

To be invited and identified, the called party has to be named. SIP chose an e-mail-like identifier of the form user@domain, user@host, user@IP_address, or phone-number@gateway because it is the most common form of user addressing in the Internet. The domain name can be either the name of the host that a user is logged onto at the time, an e-mail 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.

Using an e-mail address as a SIP address provides a scalable means by which a UAC can deliver a request

```
INVITE sip:ann@lucent.com SIP/2.0
Via: SIP/2.0/UDP 131.215.131.13;maddr=239.112.3.4;ttl=16
Via: SIP/2.0/TCP 10.0.1.1;received=128.13.44.52
From: John Smith <sip:jsmith@lucent.com>
To: Arun Netravali <sip:ann@lucent.com>
Subject: Raise
Call-ID: 132059753@mypc.domain.lucent.com
Content-Type: application/sdp
CSeq: 4711 INVITE
Content-Length: 187

v=0
o=user1 51633745 1348648134 IN IP4 128.3.4.5
s=Interactive Conference
c=IN IP4 224.2.4.4/127
t=0 0
m=audio 3456 RTP/AVP 0 22
a=rtpmap:22 application/g723.1
```

Figure 2.
Typical SIP* `INVITE` *message.

to a SIP server, which likely knows how to forward the request to the final callee—the DNS. By performing a series of DNS lookups—such as searching for service (SRV), mail exchange (MX), and address (A) records—the caller can determine the address of a server that has naming authority for all users within the domain.

The use of an e-mail-like identifier also allows SIP addresses to be easily transformed into uniform resource identifiers (URIs),[29] such as `sip:j.doe@example.com`. As such, they can be embedded in Web pages, so that clicking on the link initiates a call to that address, similar to a `mailto`[30] uniform resource location (URL) today.

Integration with Existing Protocols

One of SIP's strengths is its remarkable ability to integrate with existing protocols used on the Internet. In particular, SIP integrates well with the two dominant applications: Web and e-mail.

SIP integrates with the Web on a number of levels. First, SIP carries around MIME content, as does HTTP. This characteristic enables SIP to return Web content as a result of a call invitation. For example, a redirect response to a SIP `INVITE` can contain a HyperText Markup Language (HTML) document or a text document. This document could relate detailed information on alternate places a user might be located (including pictures and sound), or it could contain a form for submitting credit card authorization for the call. It could even return a Java* applet, which accepts input from the caller to determine where a user might be reached. As a result, SIP would integrate extremely well with Web browsers, uniting the Web and telephony to produce new, powerful services.

SIP identifies a user by means of a URL, which can be embedded either in Web pages or in e-mail, as can any other type of URL. Clicking on a URL can initiate calls, just as clicking on a Web link can access a new Web page.

One of the richest features of the Web is its programmability. Web servers can use the common gateway interface (CGI),[31] for example, to create dynamic content, customized for each user. This feature has resulted in many popular Web services, such as access to stock information and movie listings. Because SIP looks like HTTP, we have begun to explore whether CGI can be applied to Internet telephony as well. Our investigations have indicated that CGI can be applied to SIP servers, and by using backwards compatible extensions, can provide a means for rich telephony programming.[32] We have found that telephony services such as call forwarding (and all its variants), mobility, and virtual private network (VPN)—to name a few—are easily implemented with CGI. Furthermore, the wide array of software tools that exist to simplify CGI prototyping can be used for Internet telephony as well.

SIP also integrates well with e-mail and SMTP. A SIP INVITE message can be sent by e-mail when all else fails, because a SIP address is identical to an e-mail address. A SIP proxy server can easily reformat a SIP message into an SMTP message, whose formatting is nearly identical, and forward it to an SMTP MTA. This feature integrates voice mail and e-mail to enable call invitations to be delivered via e-mail when a user is not available. In addition, SIP headers can contain mailto URLs, which redirect callers to the user's e-mail, for example.

The real-time streaming protocol (RTSP)[33] allows a client to have VCR-like controls over a media server by instructing the server to play, record, fast-forward, and rewind, among other functions. The client can also specify details about the types of format in which the server should record or play. SIP also integrates well with RTSP.

Like SIP, RTSP is a textual protocol, similar in format to HTTP. RTSP has several uses in IP telephony. First, voice mail is generally considered an important part of telephony. In essence, a voice mail server is nothing more than a media server. RTSP fits naturally as a means for controlling an IP telephony voice mail server. Second, it is often useful to record a call or to play back some prerecorded content into a conference call. RTSP could be used to control the recording and playback in these cases. In fact, RTSP has been engineered to allow a client not only to invite a media server to a session using SIP, but also to control the session using RTSP.

Network Services

Because its strength is in flexible user location and name mapping features, SIP provides a good framework for services such as personal mobility; 700, 800, and 900 services; call screening; forward and transfer; and multiparty calls. (Note that billing in general is outside the scope of SIP, including 700, 800, and 900 services.) Each of these is essentially based on programmed call routing and thus fits well within SIP. Also introduced in SIP are some new services not available on the phone network, such as caller selection, described later in this paper. The sections that follow present just a few of the services possible with SIP.

Personal Mobility

SIP supports advanced personal mobility services, an example of which is shown in **Figure 3**. A user of the system, Bob, maintains an office at a Lucent Technologies location. In addition, Bob is an assistant professor at Columbia University, where he has another lab and office. Bob publishes a single IP telephony phone address for himself: bob@lucent.com. When Bob is at Columbia, he sends a REGISTER message to the Lucent SIP server (1), listing his Columbia address—bob@columbia.edu—as a forwarding address. Once at Columbia, Bob registers both his lab machine (2)—bob@lab.columbia.edu—and his office—bob@office.columbia.edu—with the Columbia registration server (3). Last time Bob was at Columbia, he set up his lab's computer to automatically forward calls to his Lucent address. Forgetting about this, Bob restarts his user agent in the lab with the same configuration.

Later in the day, jack@att.com places a call to bob@lucent.com. Using DNS, the caller resolves lucent.com to the address of the Lucent SIP server, which receives the call request (4). The server checks its registration and policy databases (5), and decides to forward the request to bob@columbia.edu. To do so, it looks up columbia.edu in DNS and obtains the address of the main Columbia SIP server. It then forwards the request there (6). As soon as the request arrives, the Columbia server looks up Bob@columbia.edu in the registration and policy database (7) and determines that he has two potential means of contact. The server then forks and sends a call request to both the lab and office machines simultaneously (8, 9), causing the office phone to ring. The lab phone receives the request, and according to its outdated configuration, forwards it to Lucent (10). Using the loop detection capabilities in SIP, the Lucent server determines that an error has occurred and returns an error response to the lab machine (11). It, in turn, returns an error code to the Columbia server (12).

In the meantime, Bob answers the phone in his office, sending an acceptance response back to the Columbia server (13). Having now received both responses, the Columbia server forwards the call acceptance back to the Lucent server (14), which forwards

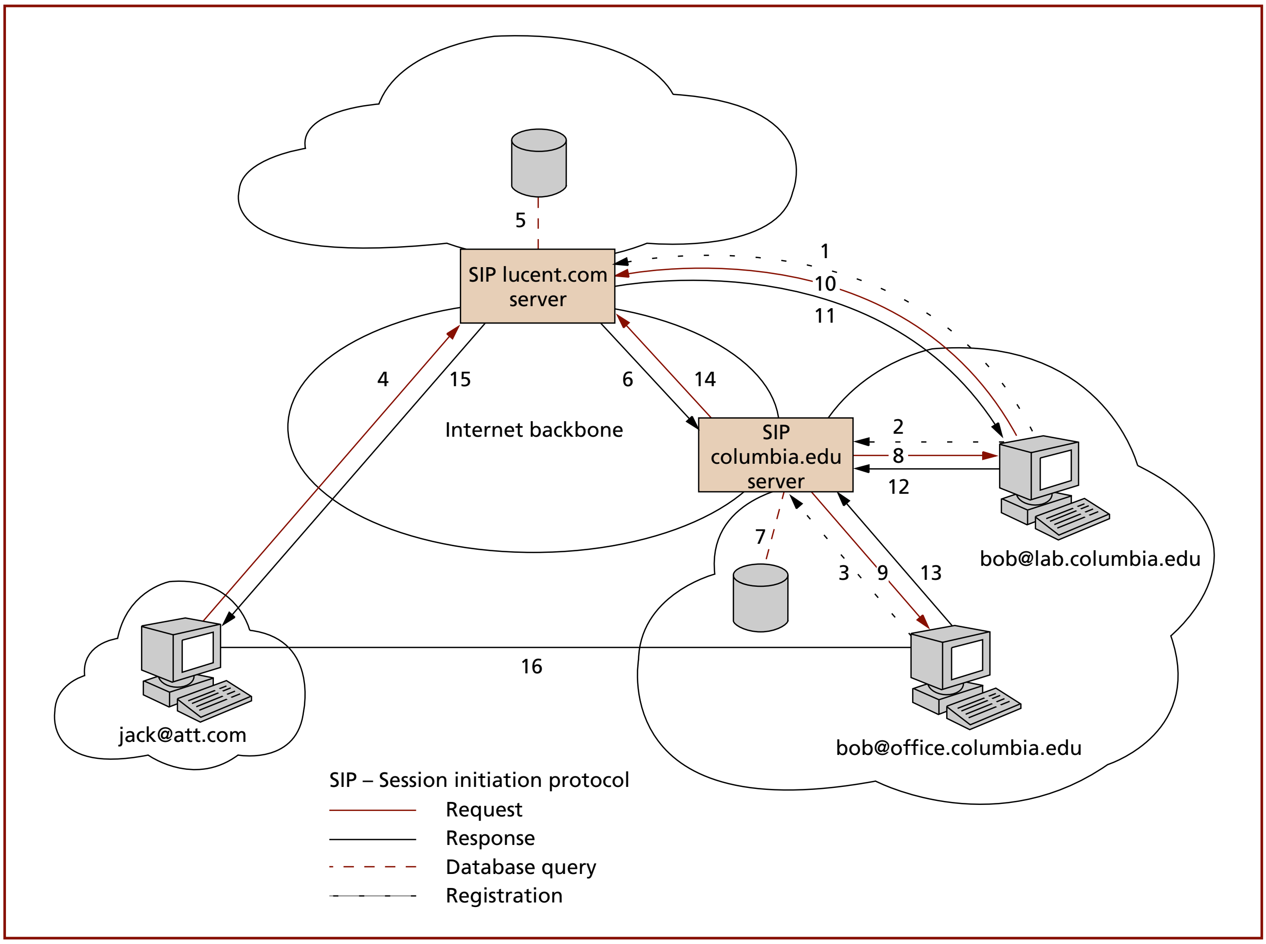


Figure 3.
An example of a SIP mobility service.

the request back to the original caller (15). At this point, the Lucent and Columbia servers can destroy all call states if they so choose. Future call transactions may proceed directly between the caller and Bob without passing through the intermediate servers (16).

The example in Figure 3 illustrates a number of salient features of SIP. First, it shows how a call request can trigger a hunt for a user, hopping among multiple servers until the final target of the call is found. Second, it demonstrates the loop detection features of SIP and, third, how a server can fork requests to speed up the process of contacting the desired user. Finally, it shows that a SIP server is normally used only for the initial call setup (although a server may elect to maintain a call state and remain on the signaling path by using SIP's source routing features). Even in this more limited role, the SIP plays an important part in the delivery of rich mobility services.

Home Phone

One of the more interesting technical challenges for IP telephony is to mimic standard residential phone service. In particular, IP telephony requires the following features:

- When someone calls a particular number, all phones in the home must ring.
- When one of the lines is picked up, all other lines must stop ringing.
- A user can pick up from any other telephone in the home and join an existing call.
- A home can have multiple lines, enabling a user on another handset to initiate a new call while one or more are in progress.
- All users involved in a single call are essentially

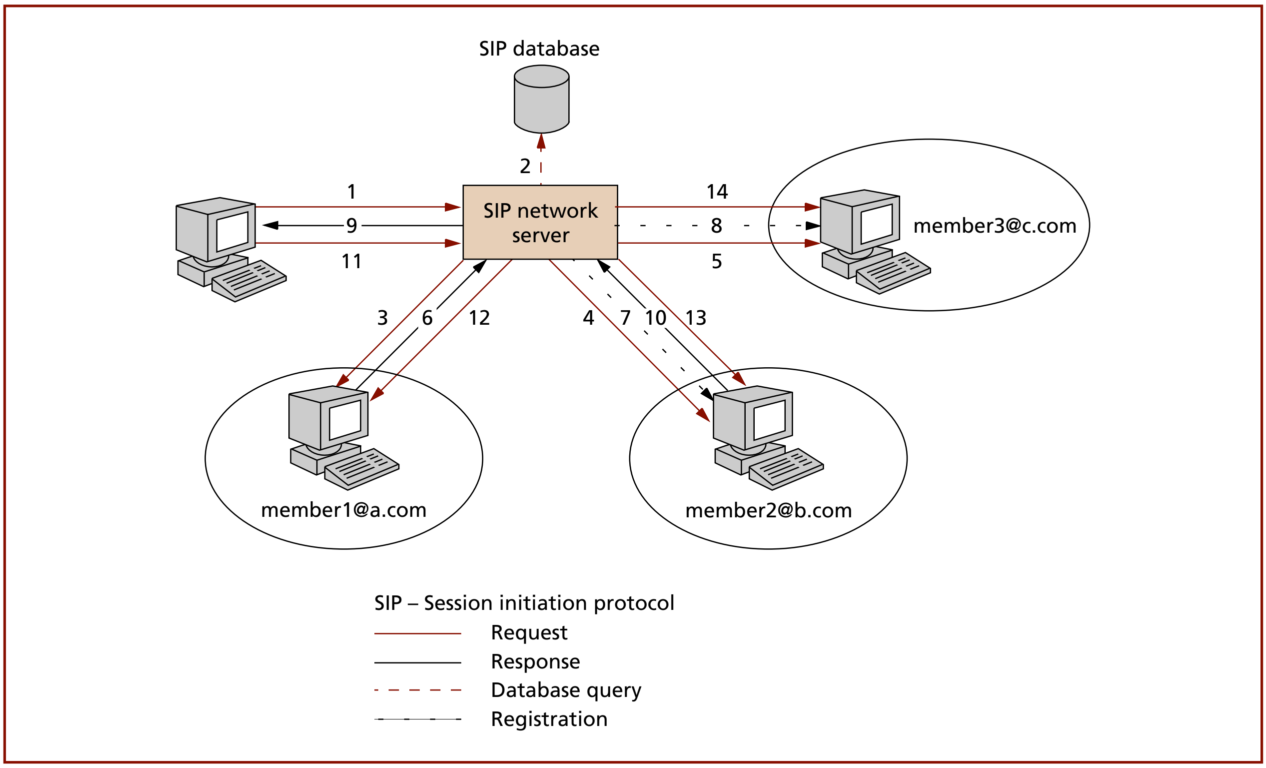


Figure 4.
SIP emulating residential service.

involved in a multiparty conference call, and are thus able to hear each other.

SIP easily emulates this basic service, as shown in **Figure 4**.

Here, the caller sends a SIP `INVITE` to `smith_family@isp.com`. The `INVITE` is sent to the main server for ISP, the Internet service provider (1). The ISP consults its policy database (2), which indicates that the called address is a residential line. The database also includes a list of contact addresses, each of which constitutes a single extension on that line. The proxy server thus forks and sends out three `INVITE` requests, (3, 4, 5), one to each address. Furthermore, the server modifies the SDP in the message, indicating that a multicast address is being used for media exchange.

The lines ring at each address. A user picks up on the first line, which causes a call acceptance to be sent back to the server (6). According to the SIP specifications, when a forking proxy server receives a call acceptance on any one of its branches, it should send a `CANCEL` request on all unanswered branches. The proxy then sends out a `CANCEL` request on the other two branches (7, 8). Because neither address has yet answered, it cancels the call request, effectively terminating the call and keeping the phones from ringing.

The server forwards the call acceptance back to the caller (9). It also acts as a multicast to unicast bridge, forwarding media received on the multicast group back to the caller, and vice versa. The server also lists itself in the call acceptance as the main point of contact for continued signaling messages.

Later, one of the other lines is picked up, forwarding another call acceptance back to the server (10). This acceptance is acknowledged by the server. Using the multicast group, the new participant is now part of the call. As soon as the caller hangs up, a `BYE` message is sent to the server (11), which in turn sends the message to all three branches (12, 13, 14). As a result, the two currently active lines terminate and the unanswered line destroys the state associated with the call.

This example further illustrates the power of forked call requests, combined with the utility of the CANCEL request. The behavior described for the SIP clients in the example described above is standard procedure, not specific for residential service. Clients of either residential or traditional business service can use the same piece of software. Only the server in the above example required special programming to direct it towards residential behavior (although this behavior was still compliant with the SIP specification). The same mechanisms can be used equally well for ACD services.

Outsourced Call Screening

The use of SIP proxy servers enables various call logic services to be outsourced to third-party vendors. Consider the case of outgoing call screening services. A parent wishes to prevent her teenage son from making calls to various adult telephone numbers. Instead of trying to keep track of the numbers of such services, and programming filters into the software in the home, the parent subscribes to a filtering service, which keeps track of such numbers automatically.

Figure 5 depicts how SIP may be used for this kind of service. The user's home software is configured to automatically forward all call requests to service831@we_do_filters.com. Whenever a call is made from the home, the call is forwarded to the server of the filtering provider (1). Once there, the service checks the URI of the incoming requests and notices that it is service 831, adult line filtering. The server performs a database lookup (2) and finds that the number called is in the database. The server then returns a call rejection to the caller (3), including a phrase that indicates this is a forbidden number. Had the called number not been in the database, the server would have proxied the call to the listed number, allowing it to complete normally.

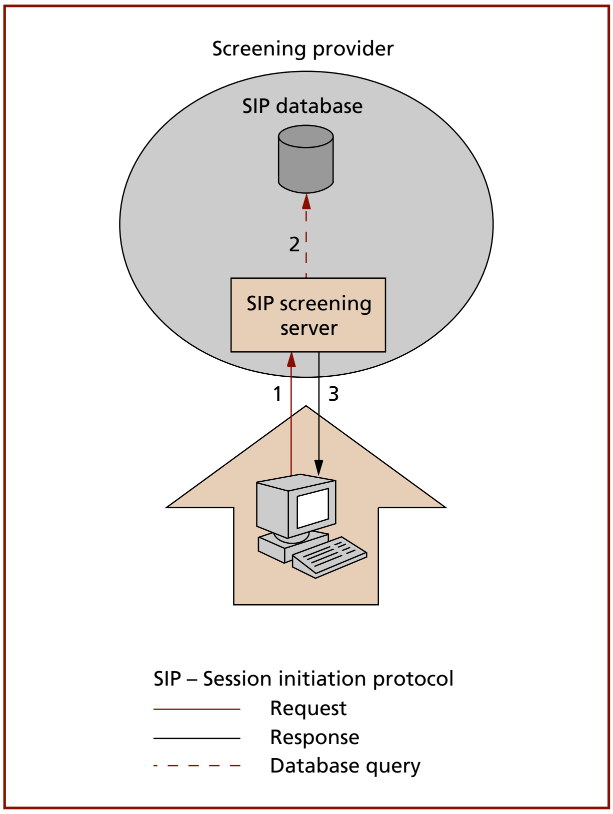


Figure 5.
SIP screening service.

Caller Selection

SIP also allows for multicast signaling and a new feature, *caller selection*, which enables the call initiator to choose whom to talk to when multiple parties answer a call. **Figure 6** demonstrates an example that uses caller selection and multicast.

In this scenario, there are no network servers. The caller, an employee of a small company, needs computer assistance, so he sends a SIP INVITE message to sysadmins@company.com. The caller's machine has been preconfigured to send messages destined for this address to a particular multicast address. All of the system administrators in the company have configured their software to listen for call requests on this address. All three system administrators receive the INVITE request (1). One of them answers immediately, and the response is also multicast (2). The software of the other system administrators reacts as if a CANCEL had been sent. Their lines stop ringing, and the call between the caller and the first administrator progresses. Later on, a second administrator answers, also sending a call acceptance on the multicast group (3). Because SIP allows multiple call acceptances to respond to a single request, the caller decides to accept the second response as well. Using SIP's third-party call control mechanisms, the caller sets up a multicast media conference among all three.

When the second call acceptance arrived, the caller could have taken any one of a number of actions. These include accepting the call, but not set-

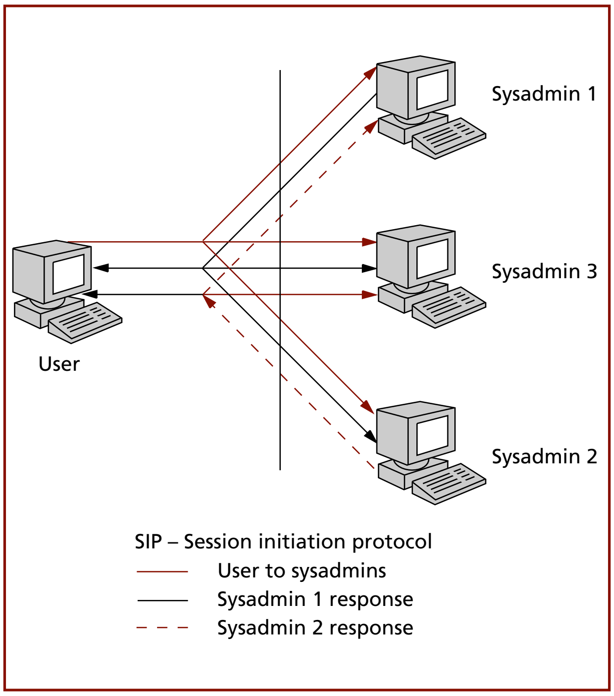


Figure 6.
An example of caller selection and multicasting.

ting up a conference (in which case the caller is effectively connected to two calls), hanging up on either call or both, or redirecting the second acceptor to voice mail. Many other possibilities exist as well. This example demonstrates a simple ACD type of service with fully distributed mechanisms. No call distribution server was required.

Client Services

As the earlier examples have illustrated, SIP network servers are generally used in services related to address resolution, naming, user location, forwarding, and policy. Telephony services encompass more than just these, of course. They include features such as multiparty conferencing, transfer, hold, and mute. An Internet environment can provide these services directly in user agent software, without the need for network server support. However, since a proxy is just a protocol server and a protocol client back to back, a network server can also provide these services using the same end-to-end mechanisms. Because the network server is a unified mechanism, it can easily deploy services at any point in the network. This type of flexibility allows SIP to be used in environments ranging from the rightly controlled and managed PBX to fully distributed, serverless networks.

SIP supports multiparty and transfer services largely through two end-to-end headers, `Also` and `Replaces`. When included in either a request or a response, these headers instruct the recipient to either place a call to the parties listed in the header or terminate it, in that order. This simple mechanism provides a framework for basic third-party call control. In turn, it may be easily used to construct a variety of transfer and multiparty services.[34]

As another example of SIP's ability to support new services, the PSTN and IP Internetworking (PINT) working group of the IETF is in the process of standardizing a protocol for click-to-dial and click-to-fax services. Click-to-dial allows a user to click on an icon while Web browsing, and receive a phone call from a service representative on the GSTN. The protocol that provides this service allows a Web server to pass third-party call control commands to a PSTN-enabled server, such as a PBX or a service control point (SCP), as shown in **Figure 7**. The requirements a protocol must fulfill to accomplish this include flexible addressing and naming, authentication, encryption, extensibility, and third-party call control primitives. Because SIP is ideal for this scenario, the working group is standardizing some basic SIP extensions to provide the service.

Key Benefits

SIP offers a number of key benefits, which we discuss here in more detail. These include simplicity, extensibility, modularity, scalability, integration, and enhanced services.

Simplicity

SIP is a very simple protocol; its specification is just 99 pages long. SIP has only 42 headers, each with a small number of values and parameters. A basic, but interoperable SIP Internet telephony implementation can get by with four headers (`To`, `From`, `Call-ID`, and `CSeq`) and three request methods (`INVITE`, `ACK`, and `BYE`), and it is small enough to be assigned as a homework programming problem in the Advanced Internet Services class taught at Columbia University. A fully functional SIP client agent with a graphical

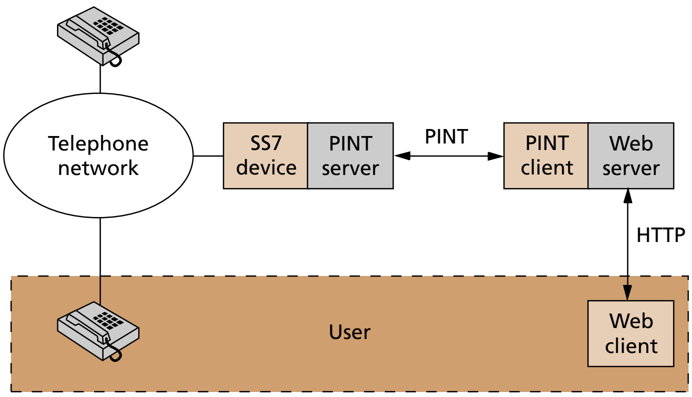


Figure 7.
Click to dial service and PINT.

user interface has already been implemented in just two months.

Because SIP encodes its messages as text, parsing and generation are simple. This is especially true when done with powerful text processing languages such as Perl. The textual encoding also simplifies debugging, allowing manual entry and perusing of messages. Its similarity to HTTP also allows for code reuse; existing HTTP parsers can be quickly modified for SIP usage.

Extensibility

Extensibility is a key metric for measuring an IP telephony signaling protocol. Currently, telephony is one of the largest, economically important, widespread, critical services, but over the long run Internet telephony is likely to supplant the existing circuit-switched infrastructure developed to support it. As with any heavily used service, the features provided evolve over time as new applications are developed, introducing problems in compatibility among versions. The Internet is an open, distributed, and evolving entity where we can expect extensions to IP telephony protocols to be widespread and uncoordinated, making it critical to build in powerful extension mechanisms from the outset.

SIP has learned the lessons of HTTP and SMTP (both of which are widely used protocols that have evolved over time), and has built in a rich set of extensibility and compatibility functions. By default, unknown headers and values are ignored. Using the `Require` header, clients can indicate named feature sets that the server must understand. When a request arrives at a server, it checks the list of named features in the `Require` header. If any of them are not supported, the server returns an error code and lists the set of features it does understand. The client can then determine the problematic feature and fall back to a simpler operation. The feature names are based on a hierarchical name space, and new feature names can be registered with the Internet Assigned Numbers Authority (IANA). SIP allows any developer to create new features within SIP and then simply register a name for them. Compatibility is still maintained across different versions.

To further enhance extensibility, numerical error codes are hierarchically organized, as in HTTP. Each of six basic classes of error codes is identified by the hundreds digit in the response code. Basic protocol operation is dictated solely by the class, and terminals need only understand the class of the response. The other digits provide usually useful, but not critical, additional information. This hierarchical model allows developers to add features by defining semantics for the error codes in a class, while maintaining compatibility.

Using textual encoding—such as `To`, `From`, and `Subject`—to describe the header fields keeps their meaning self-evident. As new header fields are added in various different implementations, developers in other corporations can determine their usage just from the name, and add support for the field. This kind of distributed, documentation-less standardization is common in SMTP, which has evolved over the years. In addition, textual encoding allows developers to add new header fields, parameters, or attributes anywhere in the message without breaking existing parsers.

Because SIP is similar to HTTP, mechanisms being developed for HTTP extensibility can also be used in SIP. Among these are the protocol extensions protocol (PEP),[35] which contains pointers to the documentation for various features within the HTTP messages themselves.

Modularity

Another aspect of extensibility is modularity. Internet telephony requires many different functions, some of which have been mentioned previously. One can be certain that mechanisms for accomplishing these functions will evolve over time (especially those that relate to QoS). It is critical to apportion these functions to separate, modular, orthogonal components, which over time can be swapped in and out of software and systems used for providing IP telephony. It is also critical to use separate, general protocols for each function, to allow it to be duplicated in other applications with ease. For example, it is more efficient to have a single application-independent QoS mechanism than it is to invent a new QoS protocol or mechanism for each application.

SIP is very modular. It encompasses basic call signaling (initiation, termination, and change), user location, and basic registration (essential for user location). Third-party call control, used for transfer and multiparty services, is in a single SIP extension. Quality of service, directory accesses, service discovery, session content description, and conference control are all orthogonal and reside in separate protocols.

A key feature of SIP is its ability to separate the notion of a session from the protocol used to invite a user to a session. SIP just issues invitations; it does not know anything about the session itself. Protocols such as SDP (which describes media sessions) are used for this purpose. SIP can also be used to invite users to control sessions, broadcast television sessions, and document editing sessions—even virtual reality sessions—once description formats for these sessions have been developed. Proxy servers do not need to know or understand anything about these sessions. For this very reason, a proxy server deployed today will still work with future applications. This kind of modularity has allowed the Web to flourish. For example, although HTTP was initially used only to carry HTML, it now carries a wide variety of content types. Had HTTP only carried HTML, there would surely be no Web.

Interestingly, the payload carried by SIP can be a program, such as a Java applet or a Tcl code fragment. These programs can dynamically define what the session is, depending on their output, making SIP a powerful tool for future intelligent network (IN) applications, where programmability is key.

Scalability

We can observe scalability on a number of different levels, including:

- *Domains*. To provide a wide area of operation, SIP leverages off DNS, as well as off powerful routing protocols such as BGP. End systems can be located anywhere on the Internet, without requiring the use of additional name resolution services.
- *Server processing*. In SIP, a transaction through several servers and gateways can be either stateful or stateless. In the case of UDP, no connection state is even required, meaning that large backbone servers can be based on UDP and operate in a stateless fashion, reducing memory requirements and improving scalability. SIP servers at network edges, or within the enterprise, can be stateful, allowing them to offer more complex services. This model of simplicity within the core and complexity at the periphery has been the cornerstone of Internet scalability for some time.

- *Conference sizes*. SIP scales to all different conference sizes. Because there is no requirement for a central multipoint controller, conference coordination can be fully distributed or centralized, at the discretion of the implementer. SIP also works for large, broadcast-style conferences. In fact, originally SIP was used to invite users to participate in Mbone conferences. For this reason, SIP is implemented in the popular mbone session directory tool, sdr, used to determine the sessions currently being broadcast on the Mbone.[36]

Integration

Another important feature of SIP is its ability to integrate well with the Web, e-mail, streaming media applications and protocols, and other networking services, to name a few. The power of the Internet is in its support for a vast array of applications. An Internet telephony protocol should therefore strive to integrate well with other Internet applications, particularly the Web.

Conclusion

In this paper, we have presented the session initiation protocol (SIP), used for the initiation, control, and termination of multimedia conferences. It builds on the strengths of HTTP and SMTP, mapping telephony services into simple request-response transactions. The strength of SIP lies in its ability to support rich mobility, forwarding, address resolution, and naming services. We have shown how SIP can be used for such things as personal mobility, call screening, residential line services, and ACD. We have briefly mentioned the third-party call control services available in SIP.

A SIP server, user agent, and gateway are all currently under development within Bell Labs. Prototypes of all three were demonstrated in June 1998 at the SUPERCOMM trade show. The SIP client is a Java-based application that uses the Java telephony application programming interface (JTAPI)[37] to rapidly develop other SIP applications. The server, a multi-platform software toolkit, incorporates support for hierarchical policy databases to guide routing decisions, which can be used to construct many of the services described in this paper. The SIP gateway interfaces to the Lucent PacketStar™ IP Services Platform, available as a new product from Lucent. The platform provides a common framework on which call services for SIP, H.323, integrated services digital network (ISDN), and "plain old telephone service" (POTS) can be constructed.

References

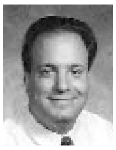
1. H. Schulzrinne, "Re-engineering the telephone system," *Proc. of IEEE Singapore Intl. Conf. on Networks (SICON) 1997: The Next Millennium*, Singapore, Apr. 14–17, 1997, pp. 261–275.
2. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications, Request for Comments (Proposed Standard) 1889," IETF, Jan. 1996.
3. B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP)—version 1 functional specification, Request for Comments (Proposed Standard) 2205," IETF, Oct. 1997.
4. B. Braden and L. Zhang, "Resource ReSerVation protocol (RSVP) version 1 message processing rules, Request for Comments (Proposed Standard) 2209," IETF, Oct. 1997.
5. P. P. Pan and H. Schulzrinne, "Yessir: A simple reservation mechanism for the Internet," *Proc. Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, U.K., July 1998.
6. S. Blake, Y. Bernet, J. Binder, M. Carlson, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss, "A Framework for Differentiated Services," Internet Draft, IETF, Oct. 1998. http://www.ietf.org/internet-drafts/draft-ietf-diffserv-framework-01.txt
7. C. Rigney, A. Rubens, W. Simpson, and S. Willens, "Remote authentication dial in user service (RADIUS), Request for Comments (Proposed Standard) 2138," IETF, Apr. 1997.
8. C. Rigney, "RADIUS accounting, Request for Comments (Informational) 2139," IETF, Apr. 1997.
9. A. Rubens and P. Calhoun, "DIAMETER base protocol," Internet Draft, IETF, May 1998. http://www.ietf.org/internet-drafts/draft- calhoun-diameter-07.txt
10. J. Rosenberg and H. Schulzrinne, "Internet telephony gateway location," *Proc. of the Conf. on Comp. Commun. (IEEE INFOCOM '98)*, San Francisco, Mar. 29–Apr. 2, 1998.
11. J. Rosenberg and H. Schulzrinne, "A Framework for a Gateway Location Protocol,"

Internet Draft, IETF, Oct. 1998. http://www.ietf.org/internet-drafts/draft-ietf-iptel-gwloc-framework-01.txt
12. T. Howes, S. Kille, and M. Wahl, "Lightweight directory access protocol (v3), Request for Comments (Proposed Standard) 2251," IETF, Dec. 1997.
13. "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service," ITU-T Rec. H.323, Geneva, May 1996.
14. M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Internet Draft, IETF, Sept. 1998. http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sip-11.txt
15. H. Schulzrinne and J. Rosenberg, "A comparison of SIP and H.323 for Internet telephony," *Proc. Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, U.K., July 1998.
16. J. Postel, "Simple mail transfer protocol, Request for Comments (Standard) STD 10, 821," IETF, Aug. 1982.
17. J. Klensin and D. Mann, "Simple mail transfer protocol," Internet Draft, IETF, May 1998. http://www.ietf.org/internet-drafts/draft-ietf-drums-smtpupd-08.txt
18. R. Fielding, J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee, "Hypertext transfer protocol HTTP/1.1, Request for Comments (Proposed Standard) 2068," IETF, Jan. 1997.
19. P. Mockapetris, "Domain names—concepts and facilities, Request for Comments (Standard) STD 13, 1034," IETF, Nov. 1987.
20. P. Mockapetris, "Domain names—implementation and specification, Request for Comments (Standard) STD 13," 1035, IETF, Nov. 1987.
21. Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4), Request for Comments (Draft Standard) 1771," IETF, Mar. 1995.
22. M. Handley and V. Jacobson, "SDP: session description protocol, Request for Comments (Proposed Standard) 2327," IETF, Apr. 1998.
23. H. Eriksson, "MBone: the multicast backbone," *Proc. of the Intl. Networking Conf. (INET)*, San Francisco, Internet Society, Aug. 1993, pp. CCC1–CCC5.
24. N. Borenstein and N. Freed, "MIME (multipurpose Internet mail extensions): Mechanisms for specifying and describing the format of Internet message bodies, Request for Comments (Proposed Standard) 1341," IETF, June 1992.
25. "Control protocol for multimedia communication," ITU-T Rec. H.245, Aug. 1997.
26. World Wide Web Consortium, "Synchronized multimedia integration language (smil) 1.0 specification, Rec. PR-smil-19980409," Apr. 1998.
27. World Wide Web Consortium, "Extensible markup language (xml) 1.0, Rec. REC-xml-19980210," Feb. 1998.
28. D. Crocker, "Standard for the format of ARPA internet text messages, Request for Comments (Standard) STD 11, 822," IETF, Aug. 1982.
29. T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource identifiers (URI): generic syntax, Request for Comments 2396," IETF, Aug. 1998.
30. P. Hoffman, L. Masinter, and J. Zawinski, "The mailto URL scheme, Request for Comments (Proposed Standard) 2368," IETF, July 1998.
31. D. Robinson and K. Coar, "The WWW common gateway interface version 1.1," Internet Draft, IETF, May 1998. http://www.ietf.org/internet-drafts/draft-coar-cgi-v11-01.txt
32. J. Lennox, J. Rosenberg, and H. Schulzrinne, "A Common Gateway Interface for SIP," Internet Draft, IETF, Nov. 1998. http://www.ietf.org/internet-drafts/draft-lennox-sip-cgi-00.txt
33. H. Schulzrinne, R. Lanphier, and A. Rao, "Real time streaming protocol (RTSP), Request for Comments (Proposed Standard) 2326," IETF, Apr. 1998.
34. H. Schulzrinne and J. Rosenberg, "Signaling for Internet telephony," *Intl. Conf. on Network Protocols (ICNP)*, Austin, Texas, Oct. 1998.
35. D. Connolly, H. Nielsen, R. Khare, and E. Prud'hommeaux, "PEP—An extension mechanism for HTTP," Internet Draft, IETF, Dec. 1997. http://www.ietf.org/internet-drafts/draft-ietf-http-pep-05.txt
36. "Session Directory," University College, London. http://www-mice.cs.ucl.ac.uk/multimedia/software/sdr/
37. Sun Microsystems, "The java telephony API." http://www.javasoft.com/products/jtapi/

(Manuscript approved December 1998)

***Trademark**
Java is a trademark of Sun Microsystems.

HENNING G. SCHULZRINNE received a B.S. degree from the Darmstadt University of Technology in Germany, an M.S. degree from the University of Cincinnati in Ohio, and a Ph.D. from the University of Massachusetts in Amherst, all in electrical engineering. An associate professor of computer science and electrical engineering at Columbia University in New York City, Dr. Schulzrinne also works as a consultant in the Wireless Networking Department at Bell Labs in Holmdel, New Jersey. His research interests include Internet telephony, Internet multimedia control and transport, and performance evaluation.



JONATHAN D. ROSENBERG, a member of technical staff in the High Speed Networks Research Department at Bell Labs in Holmdel, New Jersey, conducts research on technologies related to multimedia communications on the Internet, including transport and error recovery, signaling, architectures, protocols, and service creation. Mr. Rosenberg received B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology in Cambridge and is continuing his studies in the same field as a Ph.D. candidate at Columbia University in New York City. ◆