The Session Initiation Protocol: Internet-Centric Signaling

Henning Schulzrinne, Columbia University
Jonathan Rosenberg, dynamicsoft

ABSTRACT

The Session Initiation Protocol (SIP) provides advanced signaling and control functionality for a wide variety of multimedia services. SIP can efficiently and scalably locate resources based on a location-independent name and then negotiate session characteristics. It can find use in applications ranging from Internet telephony and conferencing to instant messaging, event notification, and the control of networked devices. We summarize the main protocol features and describe a range of extensions currently being discussed within the Internet Engineering Task Force.

INTRODUCTION

The Session Initiation Protocol (SIP) [1] initiates, modifies, and terminates network sessions. Current applications of SIP focus on interactive multimedia sessions such as Internet phone calls or multimedia conferences, but SIP or extensions of the protocol can also be used for instant messaging, event notification or managing other session types, such as distributed games. SIP was standardized by the Internet Engineering Task Force (IETF) in early 1999.

In setting up sessions, SIP acts as a signaling protocol, offering services similar to telephony signaling protocols such as Q.931 or ISUP, but in an Internet context. As we will see throughout this article, SIP also greatly extends the functionality offered by its telephony predecessors. SIP also differs from telephony signaling protocols in that it does not reserve resources or establish circuits (virtual or real) in the network.

SIP is part of the overall IETF multimedia architecture that has emerged over the past few years. This architecture includes the Real-Time Transport Protocol (RTP) for transporting audio, video and other time-sensitive data, the Real-Time Streaming Protocol (RTSP) for setting up and controlling on-demand media streams, the Media Gateway Control Protocol (MGCP) and Megaco (also known as H.248) for controlling media gateways, the Session Description Protocol (SDP) for describing multimedia sessions, the Session Announcement Protocol (SAP) for announcing multicast sessions, Telephony Routing over IP (TRIP) for locating the best gateway between the Internet and the PSTN, as well as a suite of resource management and multicast address allocation protocols.

SIP is primarily meant to set up sessions between humans identified by e-mail-like identifiers or telephone numbers, but anything addressable by a host name can participate in a SIP session. For simplicity, we will refer to SIP session participants as users.

The process of session setup involves the discovery of a user wherever located so that a description of the session can be delivered to the user. SIP itself is independent of the type or other characteristics of a session; the session description is delivered as an opaque body. (This is similar to the separation between flow specs and the resource reservation protocol, RSVP, as a delivery mechanism.)

Users can maintain the same identifier even as they change attachment points to the network or use different devices (*personal mobility*). The identifier may be assigned by their network provider, telephony service provider, or professional affiliation. (This is similar to how an e-mail recipient may retrieve messages via a variety of devices and from any network location.) In addition, a single user identity may *simultaneously* be represented by a number of network terminals. Conversely, a single network terminal or user may be reachable using multiple identifiers, again similar to e-mail. Depending on logic in SIP network elements, SIP can deliver requests to any or all of these network locations. Thus, SIP could be thought of as an application-layer anycast, with the SIP network entities performing application-layer "routing."

Session initiation also depends on the ability of the called party to have enough information about the session itself in order to make a decision on whether to join or not. Thus, SIP includes information about the caller, the purpose for the invitation, its urgency, and parameters of the session itself.

SIP was published as an IETF proposed standard (RFC 2543) in March 1999. (Proposed standard is the first standardization maturity level.) Since then, five bake-offs have taken

0163-6804/00/$10.00 © 2000 IEEE

place in intervals of four months. The April 2000 bake-off featured about 60 implementations from 45 different companies. Information about current SIP-related activities can be found at http://www.cs.columbia.edu/sip.

This article provides an overview of SIP and how it can be used to create telecommunications services as well as applications not currently supported in telephone networks. We first describe the basic protocol architecture, including how sessions are signaled, described, and secured. Then we indicate how callers can influence the routing of SIP messages. We go on to describe how SIP and quality of service (QoS) intersect. SIP also supports various forms of mobility. We describe how SIP can be used to tie together Internet telephony gateways controlled by device control protocols such as MGCP and Megaco.

One of the promises of Internet telephony is the ability to have users and administrators, rather than just equipment vendors, program services. Several options have been explored, ranging from Web-like scripting to dedicated programming languages. We conclude by summarizing where SIP may be appropriate (and where not).

The Session Initiation Protocol

Basic Architecture

Four logical types of entities participate in SIP: user agents, registrars, and proxy and redirect servers. User agents initiate requests and are usually their final destination. Internet telephones and conferencing software are examples of user agents. Registrars keep track of users within their assigned network domain (e.g., all users with identifiers *x*@`macrosoft.com` register with the registrar in the `macrosoft.com` domain). Proxy servers are application-layer routers that forward SIP requests and responses. Redirect servers receive requests and then return the location of another SIP user agent or server where the user might be found. It is quite common to find proxy, redirect, and registrar servers implemented within the same program.

In a typical SIP session, SIP messages originating at a user agent traverse one or more SIP proxy servers and then reach one or more SIP user agents. However, SIP user agents can also communicate directly with each other. Indeed, it is common that only the first request exchange travels along a chain of proxies, with all subsequent requests exchanged directly between the two user agents.

Signaling

As mentioned in the introduction, SIP's principal role is to set up sessions or associations between two (or occasionally more) Internet end systems. These associations are then used, for example, to exchange media data in an Internet phone call. The data exchange between the Internet end systems does not have to use the Internet; for example, the PINT protocol [2] uses SIP to set up calls between two regular telephones (e.g., a customer and a customer service agent).

The routing of requests is mostly determined by the request URI contained in the request. SIP request URIs look similar to e-mail addresses, consisting of a user name part and a host name part, plus a number of parameters. Indeed, it is likely that many will be able to reuse their e-mail address as a SIP address, avoiding the need to specify a different identifier for each means of communication.

Each proxy or redirect server looks at the request URI and uses it, plus any other header fields it finds useful, to route the request. Typically, the server uses backend *location servers* to map the request URI to a new destination. Location servers are logical entities whose operation is not defined by the protocol specification. (A later section describes location servers in more detail.) The request URI is then rewritten to reflect the result of this lookup process.

A user agent may decide to send all requests to a fixed, typically local, *outbound proxy* server, which then routes the request according to the request URI. In the example in Fig. 1, no outbound proxy is used. Alice, using the IP phone with the domain name `a.wonderland.com`, wants to call Bob in the `macrosoft.com` domain. Her user agents routes the request to the designated server for `macrosoft.com`, where Bob has registered his Ethernet phone as `b.macrosoft.com`. Bob is on vacation and has forwarded his calls to Carol in the same domain, who has registered at the machine `c.macrosoft.com`. The proxy server thus forwards the request there, replacing the request URI with `carol@c.macrosoft.com`. Note that the path of the signaling messages may be completely different from that of the media exchanged between caller and callee. In this respect, SIP signaling differs radically from the typical telephone signaling, where the signaling message sets up state in each intermediate telephone switch.

The host name in a SIP URI does not directly identify the physical machine. Rather, it points to a Domain Name Service (DNS) SRV record, which in turn lists one or more server addresses. These server addresses are tried in order of priority.

SIP messages can be transported on just about any communication protocol. Different protocols can be used between proxies and user agents while forwarding a single message. Generally, UDP is preferred since it avoids the TCP connection setup and teardown overhead.

SIP requests and responses are grouped into transactions. A transaction consists of one request, followed by zero or more provisional responses that indicate call progress, a final response that indicates whether the request succeeded or failed, and, for `INVITE` requests, an `ACK` request from the originator of the first request to confirm the arrival of the final response.

Responses in SIP are self-routing, tracing their way back through the same set of servers the requests visited. Each server records its address and port number in a Via header; these are then reversed by the destination.

In the example, a complete call is shown. The initial call is established by the `INVITE` request (with the `ACK`, as mentioned, for confirming call establishment). Media is then exchanged directly between the caller and the callee. Either Alice or Carol can then send a `BYE` request to tear down the session.

In a typical SIP session, SIP messages originating at a user agent traverse one or more SIP proxy servers and then reach one or more SIP user agents. However, SIP user agents can also communicate directly with each other.

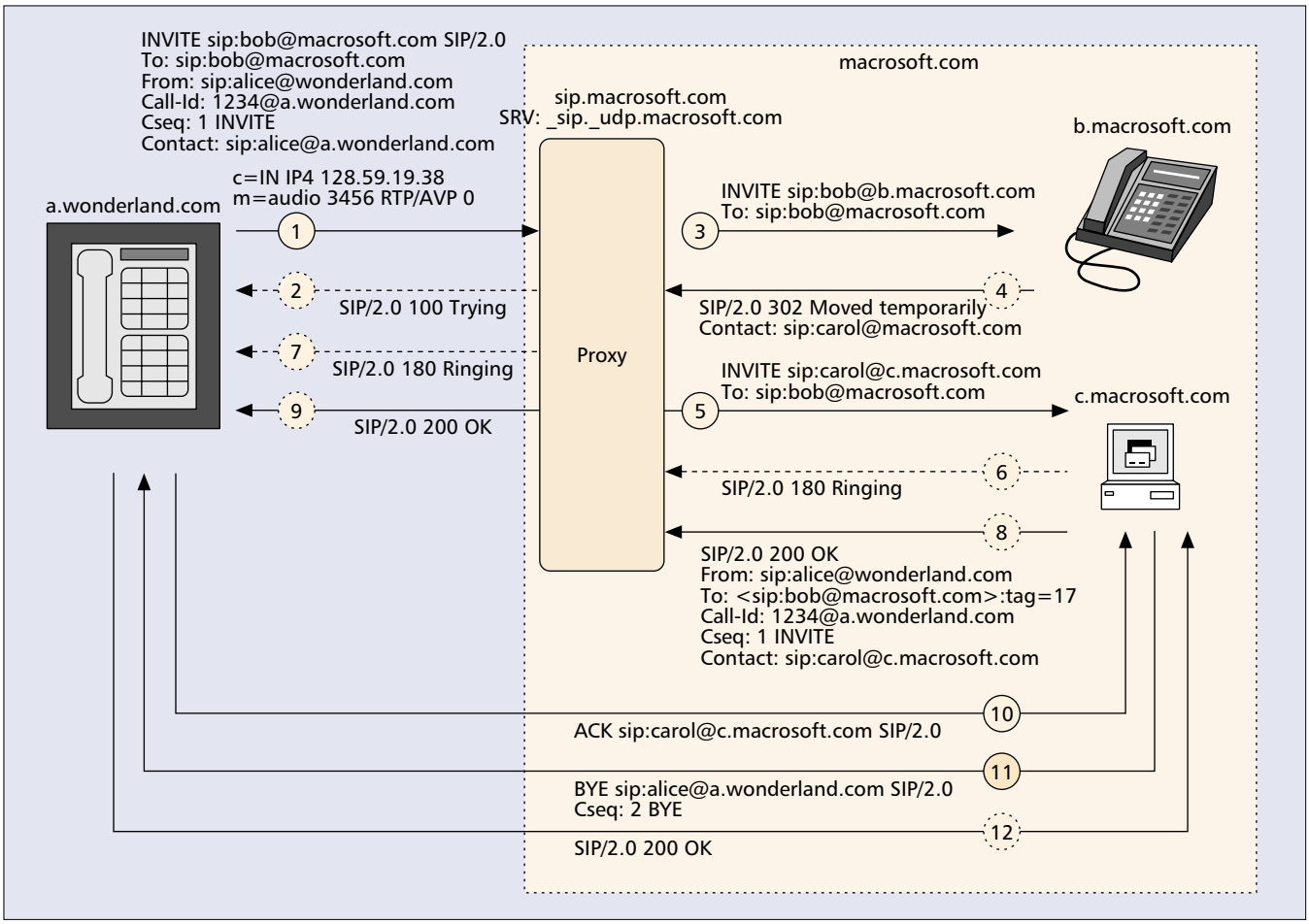

Figure 1. *An example of a forked SIP call.*

Describing and Changing Sessions

Current implementations of SIP use SDP to describe multimedia sessions. SDP is a description format that allows each party to declare what media streams it wants to receive and its receive capabilities. These streams and capabilities are expressed in a simple textual format, as shown in the message examples of Fig. 1. SDP is carried as a message body in SIP, separated by a blank line from the SIP headers. Each media stream entry indicates the destination address and port number and lists the encodings supported by the receiver, among which the sender is allowed to alternate during the session. SDP also allows to schedule sessions into the future or describe recurrent sessions.

If either party wants to add a media stream to the session, change the media destination network address, or drop a media stream, it simply sends another `INVITE` request to the other party. These requests are handled almost identically to the first one, except that the receiving user agent typically does not alert the user, and a failure in this re-`INVITE` does not tear down the call but simply leaves the old session settings in place.

SIP Message Format

SIP differs from other signaling protocols such as Q.931, ISUP, and H.323 in that it uses a textual encoding of its messages with a syntax very similar to HTTP. The use of a plain text representation is often said to incur additional message length overhead and processing costs. While, unfortunately, no formal study exists, a study [3] showed that it takes between 838 and 1240 bytes of SIP messages to set up a call, depending on whether or not text compression is used.[1]

Forking

SIP differs from other signaling protocols in that it allows a call request to *fork*; that is, a server can send out two or more requests to different destinations (*branches*) based on one incoming request, either at once or in sequence if an earlier request failed. This feature supports a number of advanced telephony services, such as call forwarding to voice mail, automatic call distribution (ACD), and user location, where the same number can ring at home and at work, for example. Typically, individual requests are generated and sent unicast to branches; however, a proxy server can also multicast a request, simplifying the building of scalable ACD systems.

Reliability

Since SIP messages can be transmitted over unreliable transport protocols like UDP, SIP has to take care of reliability on its own. It has two reliability mechanisms, one for `INVITE` requests and one for all other requests. `INVITE` requests are different since the final response (e.g., the callee picking up) can be delayed by several tens

[1] *The same study found that the same exchange would take about 800 bytes using H.323v4, a yet-to-be-standardized future version of H.323.*

of seconds from the arrival of the request. Clients retransmit `INVITE` requests until a provisional response arrives, and servers retransmit responses until confirmed by an `ACK` request. Clients retransmit other request methods until the final response arrives [4].

Locating Users

Proxy and redirect servers use a logical entity called a *location server* to route requests. The location server itself can use any method to map an incoming call to a next-hop destination. However, the primary means of locating users is by using the location mappings installed through user registrations. User agents periodically register, via the `SIP REGISTER` request, with their local registrar server, often collocated with the local proxy, indicating their current network address. Registrations can also contain additional information about the capabilities and preferences of a user, as described later.

Session Characteristics

Sessions are described at two levels: the overall characteristics and, if a media session, the session description. The overall session characteristics include the caller's name, address, and organization, the callee's name and address, the session's urgency, and its subject. The media in the session can be described in any mutually acceptable format; however, only SDP is used at this point. SDP was originally designed to describe multicast sessions, but is used in SIP for unicast sessions. SDP indicates the receive capabilities and destination addresses and ports for any number of media streams. In the example of Fig. 1, both caller and callee support only a single RTP media stream with one codec type, μ-law (codec type 0), but typically end systems offer a list of codecs that the other side can then switch between dynamically.

Extending the Protocol

Table 1 summarizes the ways SIP can be extended and the means by which the client and server can negotiate the use of the extension. In general, clients and servers can indicate their capabilities via the Supported header, enumerating the names of extensions. If a client requires that the server offer a particular feature and wants the request to be rejected otherwise, it includes this feature in a Require header.

Security

Both signaling and media need to be secured against eavesdropping and alteration. In particular, authentication is important since there is no trusted third party (the phone company) to ensure the accuracy of the information contained in the session setup request. Also, proxy servers may only want to offer services to registered users,[2] and registrations must be protected from malicious alteration.

SIP inherits the basic and digest authentication mechanisms from HTTP. Basic authentication simply requires that the sender of a request provide a plain text password. This is clearly picket-fence security, but may be acceptable if SIP messages are carried using transport-layer or IP-layer security. Digest authentication uses a challenge-response approach that checks whether the originator of a request is privy to a shared secret. In addition, SIP requests can be signed with PGP. Authentication using the CHAP challenge-response mechanism used by Point-to-Point Protocol (PPP) has been proposed.

Extension	Example	Negotiation
Method	SUBSCRIBE, NOTIFY	OPTIONS, Allow
Header	Session-Expires	Require, Supported
Body	ISUP	Accept, Accept-*
Status	183	Treat by class

Table 1. *Example SIP protocol extensions.*

SDP can convey session keys for media streams, as long as the signaling request is encrypted. Unless all proxies are trusted, only end-to-end encryption of the SIP message body can ensure confidentiality. Currently, public key cryptography using the PGP format has been defined, but any mechanism developed for e-mail should easily transfer to the SIP environment. (One difference is that part of the message needs to remain unencrypted to allow servers to forward the request appropriately.)

Integration with Other Internet Services

Internet telephony can only grow beyond a packetized replacement of the public switched telephone network (PSTN) if it integrates fully with other Internet services and protocols. SIP can be linked to other common Internet services in a variety of ways.

First, SIP addresses are regular URLs and can be embedded in Web pages, e-mail, and any other context where URLs are allowed. The request URI in a SIP request is typically a SIP URI, but could also be a `tel:` URL identifying a traditional telephone number.

In SIP, a call can be forwarded, for example, to a Web page, an e-mail address, an RTSP address for announcements, or an IRC chat channel.

The message body of requests and responses uses the MIME mechanism familiar from e-mail. SIP messages can thus carry any binary or text object, with servers treating the message body as opaque data. Typically, the message body is a session description, but it can be quite useful to include a Web page listing the callee's travel schedule, for example. The message body can also contain links to Java applets, allowing the callee to install customized user interfaces at the caller.

Telephony Service

Many traditional telephony services are provided by the baseline SIP specification described above, including caller ID, name/number mapping services (e.g., 800 and 900 services), variations on call forwarding, and putting calls on hold. Rather than enumerating dozens of services and creating protocol elements for them, it appears likely that services in the future will be generated by customized logic, as described later. SIP also attempts to provide building blocks that can be used to construct services,

Both signaling and media need to be secured against eavesdropping and alteration. In particular, authentication is important since there is no trusted third party, the phone company, to assure the accuracy of the information contained in the session setup request.

[2] *Without proxy authentication, proxies could be drafted into obscuring the caller's origin, facilitating phone spam.*

Call transfer is implemented in a decentralized fashion by having the party wishing to initiate the transfer ask the other party to invite the third party. The original call leg is then severed separately.

rather than including specific message headers or methods for each service.

Call transfer is implemented in a decentralized fashion by having the party wishing to initiate the transfer ask the other party to invite the third party. The original call leg is then severed separately.

Multiparty Sessions

While almost all phone calls are between two parties, signaling also needs to support multiparty calls. In the legacy PSTN, only centralized "bridges" (multipoint control units, MCUs) allow mixing of audio calls. SIP sessions can use three different multiparty conferencing architectures:

- **Full mesh**: In a full mesh, every participant builds a signaling leg with every other participant and sends an individual copy of the media stream to the others. This mechanism only scales to very small groups, such as three-way calls, but requires no network support. Proposals for setting up and tearing down SIP call-leg meshes have been made, but details of how each party can add other parties remain to be worked out.
- **Mixer**: A mixer or bridge takes several media streams and replicates them to all participants. Typically, audio is mixed, while video is either simply copied or combined into a single image. Participants either call into the mixer or are called by the mixer; both mechanisms are readily supported in SIP, without protocol additions.
- **Network-layer multicast**: Neither full mesh nor mixers scale to large conferences. These are most efficiently supported by network-layer multicast. Inviting a person to a multicast session is no different than any other invitation.

Expressing Caller Preferences

In the legacy PSTN, the caller only has limited choices to make. The type of communications medium or whether to reach a home or business phone, for example, is generally given by the telephone number. In SIP, individuals will likely unify many of their communications tools and location under a single public identifier. This greatly eases the space crunch on business cards, provides permanence of identifiers, and allows easy addition of new means of communications. However, some means for the caller to identify properties of the communication mechanism are desirable. For phone calls and conferences, the media description already expresses such capabilities and preferences, but a caller may, for example, want to avoid ringing a home phone number or a cellular phone, might prefer to talk to voicemail or a sales agent speaking her language, or may only want to talk to the person addressed rather than another individual to whom the callee is forwarding calls. (The latter offers the equivalent of person-to-person calls in the legacy PSTN.)

A SIP extension known as caller preferences tries to address this issue. Users register their terminal characteristics with the local proxy. The registration below indicates, for example, that Carol has access to full-duplex audio, video, and chat; speaks English, Spanish, and German; but only wants this address to be used for urgent matters.

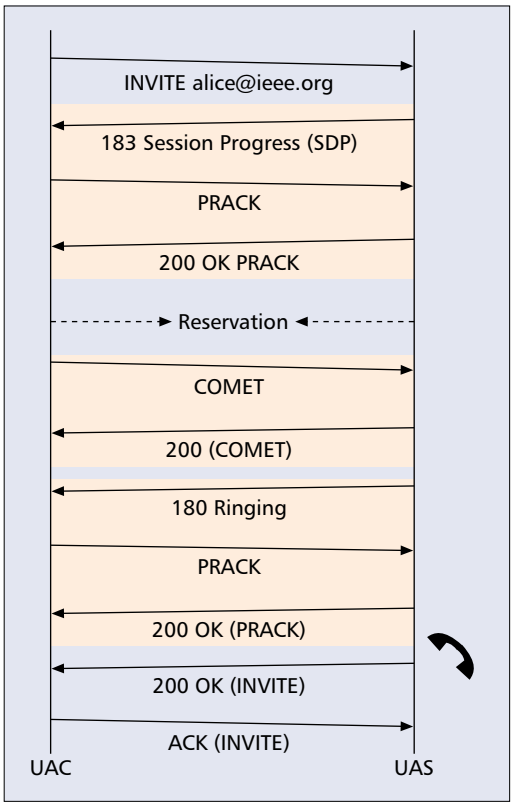

Figure 2. *A message flow diagram for interleaving resource reservation and signaling.*

```
REGISTER example.com SIP/2.0
From: Carol <sip:carol@example.com>
To: Carol <sip:carol@example.com>
Contact: Carol
 <sip:carol@host17.example.com>
 ;language="en,es,de"
 ;media="audio,video,application/chat"
 ;duplex="full"
 ;priority="urgent"
 ...
```

The caller expresses her preferences in an Accept-Contact header. For example,

```
Accept-Contact:
    sip:example.com;language="en,de"
```

indicates that the caller would prefer to somebody who speaks either English or German, and that only addresses within `example.com` are acceptable.

Other mechanisms allow explicitly ruling out forwarding to specific locations or types of destinations and restricting the type of forwarding or proxying performed by intermediate servers. Clearly, the caller is at the mercy of the intervening proxies to heed these instructions.

Quality of Service

As indicated earlier, SIP messages and data streams are likely to traverse very different parts of the network, so using SIP to directly set up resource reservations, for example, is not appro-

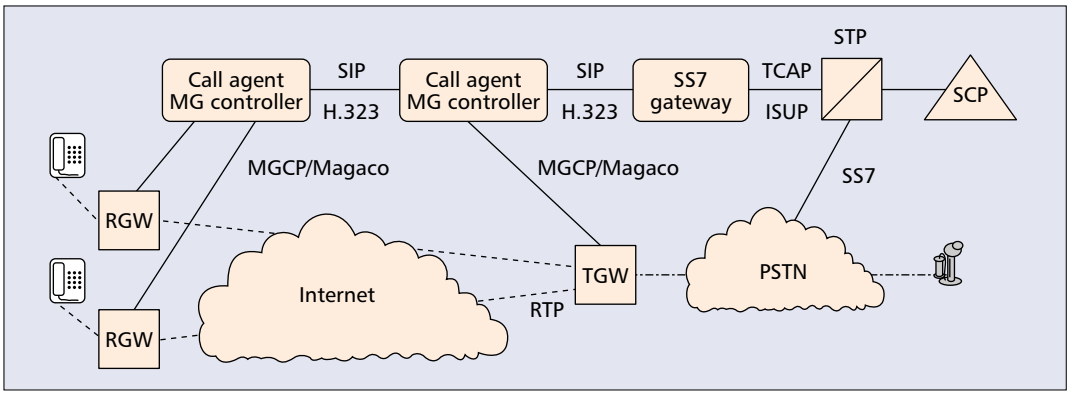

Figure 3. *Interaction of SIP and MGCP/Megaco.*

priate. However, SIP can be used to negotiate the use of QoS mechanisms. This is particularly important if the behavior of the legacy PSTN is to be replicated where the callee's phone does not ring unless a media path from caller to callee can be acquired. Without additional mechanisms, a SIP call would ring, the callee pick up, and then the line be "dead" as resource reservation fails. The caller does not generally know the callee's IP address, so it has to wait at least until the first provisional response is returned by the callee. To overcome this problem, SDP is enhanced with the ability to indicate, for each media stream, whether resource reservation is suggested or required. The caller then signals (Fig. 2) successful completion of resource reservation using the COMET request. Once the callee has completed its resource reservation successfully, it can then start ringing. This approach also works if both sides use only local resource reservation, as might be the case for reserving transmission slots on a cable modem.

Mobility

One of the central tasks of SIP is to locate one or more IP addresses where a user can receive media streams, given only a generic, location-independent address identifying a domain. This personal mobility allows a user to change communications devices without making the caller aware of these details. This mechanism makes it easy to offer precall terminal mobility, but does not directly support mid-call mobility (i.e., changing network attachment points once a session is in progress). Standard mobile IP can be used to hide the change of IP addresses during calls, but address filtering and dog-legged routing are of particular concern for latency-sensitive voice calls [5]. Another possibility is to use mid-session location updates using SIP. As long as media tools can change destination addresses in mid-session, this requires no further cooperation from the network. However, it does not work for long-lived TCP connections that may be part of a session (e.g., a chat session).

SIP and MGCP/Megaco

PSTN gateways need to fulfill three rather different functions: compress and decompress large volumes of audio traffic (and perform other signal processing tasks, e.g., tone detection and echo cancellation), translate between PSTN and Internet signaling, and provide services. Audio streams are often handled by digital signal processors (DSPs), even general-purpose or specifically engineered for speech coding. But DSPs are not well suited to implementing protocol stacks and large volumes of general-purpose code.[3]

Thus, it has been proposed to decompose the PSTN gateway functionality into a media gateway, a signaling gateway, and a media controller [7]. First MGCP [8] and then Megaco/H.248 have been proposed as protocols to connect the media controller with the media gateway.

However, media gateways are limited to operation within a single domain, since they require a single controlling entity, the media gateway controller (MGC). Thus, it is generally assumed that a peer-to-peer protocol is needed to connect MGCs. In addition, it is desirable that gateways be able to directly communicate with intelligent end systems, such as PC-based conferencing applications, which are not well suited to gateway decomposition. SIP has been proposed to serve this role, as shown in Fig. 3. The figure shows, on the left, two phones connected through residential gateways (RGWs) which act as media gateways. On the right, a large trunking gateway (TGW) is connected to the PSTN. The Signaling System No. 7 (SS7) signaling from the PSTN is tunneled from the SS7 gateway to the MGC. Assuming a user on one of the phones on the left wishes to make a call to the PSTN, the left MGC initiates SIP call signaling with the right MGC in order to complete the call. Both MGCs would use MGCP/Megaco to control their respective media gateways.

Beyond basic call establishment, there are scenarios where a PSTN gateway connects to another PSTN gateway, effectively using IP as a long distance transport mechanism. It is desirable to make the IP portion of the call as transparent to the legacy PSTN as possible. In particular, some features of ISUP do not have direct SIP equivalents (and may not make sense in a purely Internet-based environment). For these, it has been proposed to carry ISUP messages as MIME body parts in SIP messages and to add an additional SIP method for transporting mid-call ISUP messages that do not affect call state.

One of the central tasks of SIP is to locate one or more IP addresses where a user can receive media streams, given only a generic, location-independent address identifying a domain. This allows a user to change communications devices without making the caller aware of these details.

[3] *For somewhat different reasons, MGCP/Megaco has also been proposed for use in so-called residential gateways, typically cable or DSL modems. Stimulus control protocols move almost all services into the realm of the service provider [6].*

In order to minimize concerns about security and server resource consumption, CPL intentionally does not offer traditional programming language constructs such as I/O and loops, limiting its generality.

Programming SIP Services

Traditionally, telephony services have been either hard-coded into switches, made available by application programming interfaces (APIs) such as the Telephony API (TAPI) or Java TAPI (JTAPI), or made configurable in limited ways by proprietary service creation environments, in the context of the intelligent network (IN) notion of trigger points. For SIP-based telephony there are at least four choices to create SIP-based services [9]. First, language-based APIs such as Parlay and Jain allow Internet telephony to be treated similar to traditional phone calls. However, because they need to provide a uniform interface to legacy phone calls and Internet signaling, it is hard to access all the capabilities of the latter. Second, we can create dedicated languages, such as the Call Processing Language (CPL), geared to creating session services. CPL, for example, is generally installed in proxy servers and can determine how `SIP INVITE` transactions are handled (i.e., whether an incoming request should be rejected, forwarded, or proxied). This behavior can depend on the time of day or the value of any header field. (CPL is largely independent of the signaling mechanism, but has been implemented primarily in SIP servers so far.) In order to minimize concerns about security and server resource consumption, CPL intentionally does not offer traditional programming language constructs such as I/O and loops, limiting its generality.

For Web servers, two additional server-side approaches have been used to create services: common gateway interface (cgi) and Java servlets. Both approaches can also be used for SIP servers [10, 11]. Unlike HTTP cgi scripts, sip-cgi scripts can generate multiple responses and additional requests (e.g., to proxy an incoming SIP request to multiple destinations) and ask to be reinvoked on future requests or responses within the same transaction. Sip-cgi uses operating system means, standard input, and environment variables, to convey information about incoming requests. Thus, sip-cgi is the only method of the four discussed here which is independent of the programming language. The server simply invokes either an interpreter for a scripting language or a compiled binary and exchanges information with that process. A SIP servlet is a piece of Java code that receives calls from the SIP server and instructs the SIP server how to handle requests.

A fourth mechanism includes Java applets in SIP requests [12], similar to mobile agents. It allows the user agent to control delivery of services and forking, but the interaction between the goals of the sender of the request (e.g., the caller in an `INVITE`) and the receiver (callee) remain to be worked out.

Sip-cgi and CPL scripts or servlets must somehow find their way into the right server. One possibility is to have the client upload the scripts or servlet code via `REGISTER`, but this requires that the user agent store this information. Only SIP user agents implemented on workstations and PCs are likely to have this capability. For others, a separate mechanism (e.g., based on Web upload) may be preferable.

It is interesting to note that sip-cgi, CPL, and mobile code can be considered examples of active networks at the application layer. The first two preinstall request handling logic, while mobile code carries it in the request itself. (Caller preferences, described earlier, can also be considered an extremely simplified version of mobile code.)

All four mechanism need to address how features interact. The problem of feature interaction has been studied extensively in the context of traditional telephony services, but the creation of services by "laypeople," the lack of a PSTN-like trust model, and their more distributed nature raise additional issues [13].

The Applicability of SIP

There are many functions SIP explicitly does not provide. It is not a session management or conference control protocol. The particulars of the communications within the session are outside of SIP. This includes features such as media transport, voting and polling, virtual microphone passing, chair election, floor control, and feedback on session quality.

SIP is not a resource reservation protocol for sessions since SIP is independent of the underlying session it establishes, and the path of SIP messages is completely independent of the path packets for a session may take. Path independence refers to paths within a provider's network and the set of providers itself. For example, it is perfectly reasonable for a SIP message to traverse a completely different set of autonomous systems than the audio in a voice-over-IP call SIP establishes.

SIP is not a transfer protocol. It is not meant to send large amounts of data unrelated to SIP's operation, nor to replace HTTP. This is for numerous reasons, one of which is that SIP's recommended mode of operation is over UDP. Sending large messages over SIP can lead to fragmentation at the IP layer and thus poor performance in even mildly lossy networks. This is not to say that carrying payloads in SIP messages is never a good thing; in many cases, the data is very much related to the operation of SIP.

Related Work

The notion of controlling a data stream with a control stream dates back to 1972, with ftp. Telephony signaling protocols have progressed from channel-associated signaling to out-of-band signaling. Recent work known as Q.BICC tries to allow ISUP to set up connections between IP endpoints. The International Telecommunications Union (ITU) first published the H.323 suite of protocols, including the ISDN-derived signaling mechanism, in 1996 and has updated the specification since then.

SIP is related to messaging mechanisms such as Internet e-mail. For example, it shares the ability to deliver information without having to know the precise network location of the recipient. However, SIP messaging is synchronous, with end-to-end notification of success or failure, and designed for subsecond delivery times.

The ICEBERG project also considers setup of large-scale sessions using multicast [14], while Elliott [15] discusses session membership propagation.

ACKNOWLEDGMENTS

A large number of individuals contribute to the development of SIP within the MMUSIC and SIP working groups in the IETF. The RFCs and Internet drafts should be consulted for proper attribution.

REFERENCES

[1] M. Handley *et al.*, "SIP: Session Initiation Protocol," RFC 2543, Mar. 1999.
[2] S. Petrack and L. Conroy, "The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services," RFC 2848, June 2000.
[3] Nortel Networks, "A Comparison of H.323v4 and SIP." 3GPP cont., Jan. 2000.
[4] T. Eyers and H. Schulzrinne, "Predicting Internet Telephony Call Setup Delay," *Proc. 1st IP-Telephony Wksp.*, Berlin, Germany, Apr. 2000.
[5] E. Wedlund and H. Schulzrinne, "Mobility Support Using SIP," *2nd ACM/IEEE Int'l. Conf. Wireless and Mobile Multimedia*, Seattle, WA, Aug. 1999.
[6] C. Huitema *et al.*, "An Architecture for Internet Telephony Service for Residential Customers," *IEEE Network*, vol. 13, May/June 1999, pp. 50–57.
[7] N. Greene, M. Ramalho, and B. Rosen, "Media Gateway Control Protocol Architecture and Requirements," RFC 2805, Apr. 2000.
[8] M. Arango et al., "Media Gateway Control Protocol (MGCP) v. 1.0," RFC 2705, Oct. 1999.
[9] J. Rosenberg, J. Lennox, and H. Schulzrinne, "Programming Internet TELEPHONY services," *IEEE Network*, vol. 13, May/June 1999, pp. 42–49.
[10] J. Lennox, J. Rosenberg, and H. Schulzrinne, "Common Gateway Interface for SIP," Internet draft, May 1999; work in progress.
[11] A. Kristensen, A. Byttner, and R. Kurmanowytsch, "Programming SIP Services," *Proc. 1st IP Telephony Wksp.*, Berlin, Germany, Apr. 2000.
[12] M. O'Doherty, "Java Enhanced SIP (JES)," Internet draft, Mar. 2000; work in progress.
[13] J. Lennox and H. Schulzrinne, "Feature Interaction in Internet Telephony," *Proc. Feature Interaction in Telecommun. and Software Sys. VI*, Glasgow, United Kingdom, May 2000.
[14] H. J. Wang, A. D. Joseph, and R. H. Katz, "A Signaling System Using Lightweight Call Sessions," *Proc. IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000.
[15] C. Elliott, "A 'Sticky' Conference Control Protocol," *Internetworking: Research and Experience*, vol. 5, 1994, pp. 97–119.

BIOGRAPHY

HENNING SCHULZRINNE received his undergraduate degree in economics and electrical engineering from Darmstadt University of Technology, Germany, his M.S.E.E. degree as a Fulbright scholar from the University of Cincinnati, Ohio, and his Ph.D. degree from the University of Massachusetts, Amherst. He was a member of technical staff at AT&T Bell Laboratories, Murray Hill, New Jersey, and an associate department head at GMD-Fokus, Berlin, Germany, before joining the Computer Science and Electrical Engineering departments at Columbia University, New York. His research interests encompass real-time multimedia network services in the Internet, and modeling and performance evaluation. He is an editor of the *Journal of Communications and Networks*, *IEEE Transactions on Image Processing*, and IEEE Communications Society editor of *IEEE Internet Computing*. He co-chairs the IEEE Communications Society Internet Technical Committee and is chair of the IEEE Communications Society Technical Committee on Computer Communications. He is also technical co-chair of INFOCOM 2000. He is currently serving as a member of the Internet Architecture Board (IAB).

JONATHAN ROSENBERG (jdrosen@dynamicsoft.com) is currently chief scientist for dynamicsoft, Inc., a startup company in New Jersey focused on delivering infrastructure solutions for communications service providers and ASPs. He is responsible for guiding the technology directions for dynamicsoft's product line. His research interests include multimedia communications over packet networks, including signaling, transport, services, and architectures. Prior to joining dynamicsoft, he was a member of technical staff at Bell Laboratories, conducting research in these areas. He received B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, and is continuing his studies in the same field as a Ph.D. candidate at Columbia University, New York City. He is active in the IETF, where he chairs the IP Telephony working group. He is an author of numerous IETF RFCs, including SIP, RFC 2543.

SIP is not a transfer protocol. It is not meant to send large amounts of data unrelated to SIPs operation. It is not meant as a replacement for HTTP. This is for numerous reasons, one of which is that SIP's recommended mode of operation is over UDP.