Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-mmusic-sip-03.ps

MMUSIC WG
Handley/Schulzrinne/Schooler
ISI/Columbia U./Caltech

July 31, 1997
Expires: January 20, 1998

SIP: Session Initiation Protocol

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress”.

To learn the current status of any Internet-Draft, please check the “1id-abstracts.txt” listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited.

Abstract

Many styles of multimedia conferencing are likely to co-exist on the Internet, and many of them share the need to invite users to participate. The Session Initiation Protocol (SIP) is a simple protocol designed to enable the invitation of users to participate in such multimedia sessions. It is not tied to any specific conference control scheme. In particular, it aims to enable user mobility by relaying and redirecting invitations to a user’s current location.

This document is a product of the Multi-party Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group’s mailing list at confctrl@isi.edu and/or the authors.

Contents

1 Introduction

1.1 Overview of SIP Functionality

The Session Initiation Protocol (SIP) is an application-layer protocol that can establish and control multimedia sessions or calls. These multimedia sessions include multimedia conferences, distance learning, Internet telephony and similar applications. SIP can initiate both unicast and multicast sessions; the initiator does not necessarily have to be a member of the session. Media and participants can be added to an existing session. SIP can be used to "call" both persons and "robots", for example, to invite a media storage device to record an ongoing conference or to invite a video-on-demand server to play a video into a conference. (SIP does not directly control these services, however; see RTSP [4].)

SIP transparently supports name mapping and redirection services, allowing the implementation of telephony services such as selective call forwarding, selective call rejection, conditional and unconditional call forwarding, call forwarding busy, call forwarding no response. SIP may use multicast to try several possible callee locations at the same time.

SIP supports *personal mobility*. In the parlance of telecommunications intelligent network services, this is defined as: "Personal mobility is the ability of end users to originate and receive calls and access subscribed telecommunication services on any terminal in any location, and the ability of the network to identify end users as they move. Personal mobility is based on the use of a unique personal identity (i.e., 'personal number')." [1, p. 44]. Personal mobility complements terminal mobility, i.e., the ability to maintain communications when moving a single end system from one network to another.

SIP supports some or all of four facets of establishing multimedia communications:

1. user location: determination of the end system to be used for communication;
2. user capabilities: determination of the media and media parameters to be used;
3. user availability: determination of the willingness of the called party to engage in communications;
4. call setup ("ringing", establishment of call parameters at both called and calling party)

In particular, SIP can be used to locate a user and determine the appropriate end system, leaving the actual call establishment to other protocols such as H.323.

SIP may also be used to terminate and transfer a call. SIP can also initiate multi-party calls using a multipoint control unit (MCU) or fully-meshed interconnection instead of multicast.

These features are for further study.

SIP is not a conference control protocol, but can be used to introduce conference control protocols to a session.

SIP is designed as part of the overall IETF multimedia data and control architecture currently incorporating protocols such as RSVP [2] for reserving network resources, RTP [3] for transporting real-time data and providing QOS feedback, RTSP [4] for controlling delivery of streaming media, SAP [5] for advertising multimedia sessions via multicast and SDP [6] for describing multimedia sessions, but SIP does not depend for its operation on any of these protocols.

1.2 Finding Multimedia Sessions

There are two basic ways to locate and to participate in a multimedia session:

Advertisement: The session is advertised, potential participants see the advertisement, then join the session address to participate.

Invitation: Users are invited by others to participate in a session, which may or may not be advertised.

Sessions may be advertised using multicast protocols such as SAP [5], electronic mail, news groups, web pages or directories (LDAP), among others. SIP serves the role of the invitation protocol.

SIP does not prescribe how a conference is to be managed, e.g., whether it uses a central server to manage conference and participant state or distributes state via multicast.

SIP does not allocate multicast addresses, leaving this functionality to protocols such as SAP [5].

SIP can invite users to conferences with and without resource reservation. SIP does not reserve resources, but may convey to the invited system the information necessary to do this. Quality-of-service guarantees, if required, may depend on knowing the full membership of the session; this information may or may not be known to the agent performing session invitation.

SIP offers some of the same functionality as H.323, but may also be used in conjunction with it. In this mode, H.323 is used to locate the appropriate terminal identified by a H.245 address [TBD: what does this look like?]. An H.323-capable terminal then proceeds with a normal H.323/H.245 invitation [7].

1.3 Terminology

In this document, the key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in RFC 2119 [8] and indicate requirement levels for compliant SIP implementations.

1.4 Definitions

This specification uses a number of terms to refer to the roles played by participants in SIP communications. The definitions of client, server and proxy are similar to those used by the Hypertext Transport Protocol (HTTP) [9].

Client: An application program that establishes connections for the purpose of sending requests. Clients may or may not interact directly with a human user.

Final response: A response that terminates a →SIP transaction, as opposed to a →*provisional response*. All 2xx, 3xx, 4xx, and 5xx responses are final.

Initiator, calling party: The party initiating a conference invitation. Note that the calling party does not have to be the same as the one creating a conference.

Invitation: A request sent to a user (or service) requesting participation in a session.

Invitee, invited user, called party: The person or service that the calling party is trying to invite to a conference.

Location server: A program that is contacted by a →client and that returns one or more possible locations of the called party or service. Location servers may be invoked by SIP redirect and proxy servers and may be Co-located with a SIP server.

Location service: A service used by a →redirect or →proxy server to obtain information about a callee's possible location.

Provisional response: A response used by the server to indicate progress, but that does not terminate a →SIP transaction. All 1xx and 6xx responses are provisional. Other responses are considered →final.

Proxy, proxy server: An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy must interpret, and, if necessary, rewrite a request message before forwarding it.

Redirect server: A server that accepts a SIP request, maps the address into zero or more new addresses and returns these addresses to the client. Unlike a →proxy server, it does not initiate its own SIP request.

Server: An application program that accepts requests in order to service requests and sends back responses to those requests. Servers are either proxy, redirect or user agent servers. An application program may act as both server and client.

Session: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session." [6] For SIP, a session is equivalent to a "call". (Note: a session as defined here may comprise one or more RTP sessions.)

(SIP) transaction: A SIP transaction occurs between a →client and a →server and comprises all messages from the first request sent from the client to the server up to a →final (non-1xx) response sent from the server to the client. A transaction is for a single call (identified by a Call-ID, Section 6.11). There can only be one pending transaction between a server and client for each call id.

User agent server, called user agent: The server application that contacts the user when a session request is received and that returns a reply on behalf of the user. The reply may accept, reject or redirect the call. (Note: in SIP, user agents can be both clients and servers.)

An application program may be capable of acting both as a client and a server. For example, a typical multimedia conference control application would act as a client to initiate calls or to invite others to conferences and as a user agent server to accept invitations.

1.5 Protocol Properties

1.5.1 Minimal State

There is no concept of an ongoing SIP session that lasts for the duration of the conference or call. Rather, a single conference session or call may involve one or more SIP request-response transactions. For example, a conference control protocol may use SIP to add or remove a media stream, but again, once the information has been successfully conveyed to the participants, SIP is then no longer involved.

At most, a server has to maintain state for a single SIP transaction. In some cases, it can process each message without regard to previous messages (*stateless server*), as described in Section 12.

1.5.2 Transport-Protocol Neutral

SIP is able to utilize both UDP and TCP as transport protocols. UDP allows the application to more carefully control the timing of messages and their retransmission, to perform parallel searches without requiring connection state for each outstanding request, and to use multicast. TCP allows easier passage through existing firewalls, and given the similar protocol design, allows common servers for SIP, HTTP and the Real Time Streaming Protocol (RTSP) [4].

When TCP is used, SIP can use one or more connections to attempt to contact a user or to modify parameters of an existing session. The concept of a session is not implicitly bound to a TCP connection, so the initial SIP request and a subsequent SIP request may use different TCP connections or a single persistent connection as appropriate.

Clients SHOULD implement both UDP and TCP transport, servers MUST.

1.5.3 Text-Based

SIP is text based. This allows easy implementation in languages such as Tcl and Perl, allows easy debugging, and most importantly, makes SIP flexible and extensible. As SIP is primarily used for session initiation, it is believed that the additional overhead of using a text-based protocol is not significant.

1.6 SIP Addressing

SIP uses two kinds of address identifiers, *host-specific addresses* and *host-independent addresses*. Both forms of addresses take the form *user@host*, where *user* is any alphanumeric identifier and the form of *host* depends on the address type. Note that SIP does not distinguish between the two and can, while inviting a user, map repeatedly between the two address types.

For a host-specific address, the *user* part is an operating-system user name. The *host* part is either a domain name having a DNS A (address) record, or a numeric network address. Examples include:

```
mjh@metro.isi.edu
hgs@erlang.cs.columbia.edu
root@193.175.132.42
```

A user's host-specific address can be obtained out-of-band, can be learned via existing media agents, can be included in some mailers' message headers, or can be recorded during previous invitation interactions.

Host-independent addresses contain a moniker (such as a civil name) or user name and domain name that may not map into a single host. [1].

Host-independent addresses may use any unambiguous user name, including aliases, identifying the called party as the *user* part of the address. They may use a domain name having an MX [10], SRV [11] or A [12] record for the *host* part. These addresses may have different degrees of location- and provider-independence and are often chosen to be mnemonic. In many cases, the host-independent SIP address can be the same as a user's electronic mail address, but this is not required. SIP can thus leverage off the domain name system (DNS) to provide a first-stage location mechanisms. Examples of host-independent names include

```
M.Handley@cs.ucl.ac.uk
H.G.Schulzrinne@ieee.org
info@ietf.org
```

An address can designate an individual (possibly located at one of several end systems), the first available person from a group of individuals or a whole group. The form of the address, e.g., `sales@example.com`, is not sufficient, in general, to determine the intent of the caller.

[1] We avoid the term *location-independent*, since the address may indeed refer to a specific location, e.g., a company department.

If a user or service chooses to be reachable at an address that is guessable from the person's name and organizational affiliation, the traditional method of ensuring privacy by having an unlisted "phone" number is compromised. However, unlike traditional telephony, SIP offers authentication and access control mechanisms and can avail itself of lower-layer security mechanisms, so that client software can reject unauthorized or undesired call attempts.

1.7 Locating a SIP Server

Call setup may proceed in several phases. A SIP client MUST follow the following steps to resolve the *user* part of a callee address. If a client only supports TCP or UDP, but not both, the respective address type is omitted.

1. If there is a SRV DNS resource record [11] of type `sip.udp`, contact the listed SIP servers in order of preference value using UDP as a transport protocol at the port number listed in the DNS resource record.
2. If there is a SRV DNS resource record [11] of type `sip.tcp`, contact the listed SIP servers in order of preference value using TCP as a transport protocol at the port number listed in the DNS resource record.
3. If there is a DNS MX record [10], contact the hosts listed in their order of preference at the default port number (TBD). For each host listed, first try to contact the server using UDP, then TCP.
4. Finally, check if there is a DNS CNAME or A record for the given *host* and try to contact a SIP server at the one or more addresses listed, again trying first UDP, then TCP.
5. If all of the above methods fail, the caller MAY contact an SMTP server at the user's *host* and use the SMTP EXPN command to obtain an alternate address and repeat the steps above. As a last resort, a client MAY choose to deliver the session description to the callee using electronic mail.

If a server is found using one of the methods below, the other methods are not tried. A client SHOULD rely on ICMP "Port Unreachable" messages rather than time-outs to determine that a server is not reachable at a particular address. A client MAY cache the result of the reachability steps, but SHOULD start at the beginning of the sequence when the cached address fails.

Implementation note for socket-based programs: For TCP, `connect()` returns `ECONNREFUSED` if there is no server at the designated address; for UDP, the socket should be bound to the destination address using `connect()` rather than `sendto()` or similar.

This sequence is modeled after that described for SMTP, where MX records are to be checked before A records [13].

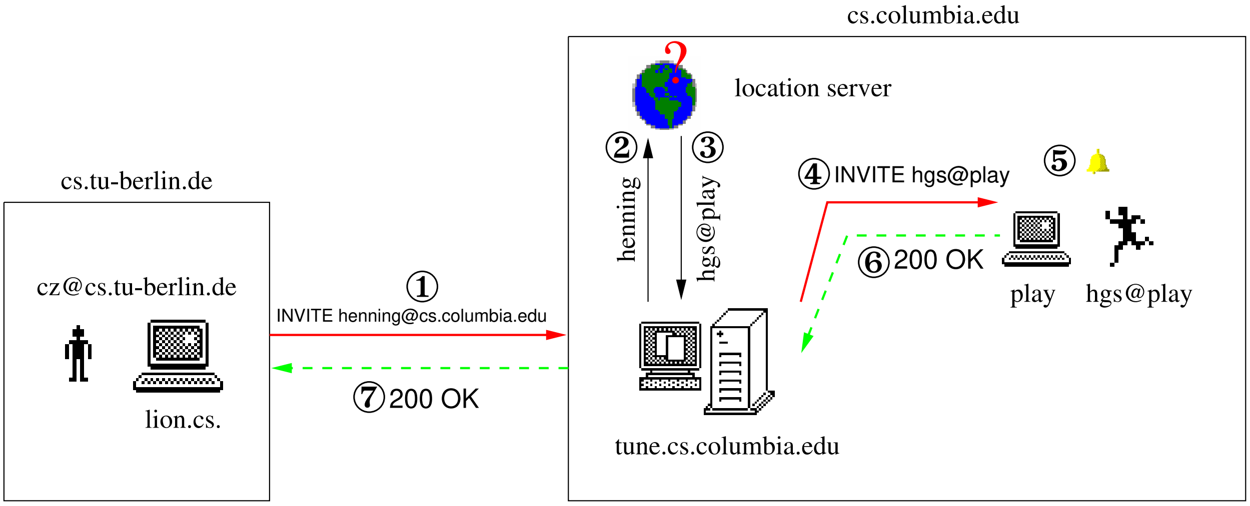

Figure 1: Example of SIP proxy server

1.8 SIP Transactions

Once the *host* part has been resolved to a SIP server, the client sends one or more SIP requests to that server and receives one or more responses from the server. If the invitation is SIP request is an invitation, it contains a session description, for example written in SDP format, that provides the called party with enough information to join the session.

If TCP is used, request and responses within a single SIP transaction are carried over the same TCP connection. Thus, the client SHOULD maintain the connection until a final response has been received. Several SIP requests from the same client to the same server may use the same TCP connection or may open a new connection for each request. If the client sent the request sends via unicast UDP, the response is sent to the source address of the UDP request. If the request is sent via multicast UDP, the response is directed to the same multicast address and destination port. For UDP, reliability is achieved using retransmission (Section 11).

Need motivation why we ALWAYS want to have a multicast return.

The SIP message format and operation is independent of the transport protocol.

The basic message flow is shown in Fig. 1 and Fig. 2, for proxy and redirect modes, respectively.

1.9 Locating a User

A callee may move between a number of different end systems over time. These locations can be dynamically registered with a location server, typically for a single administrative domain, or a location server may

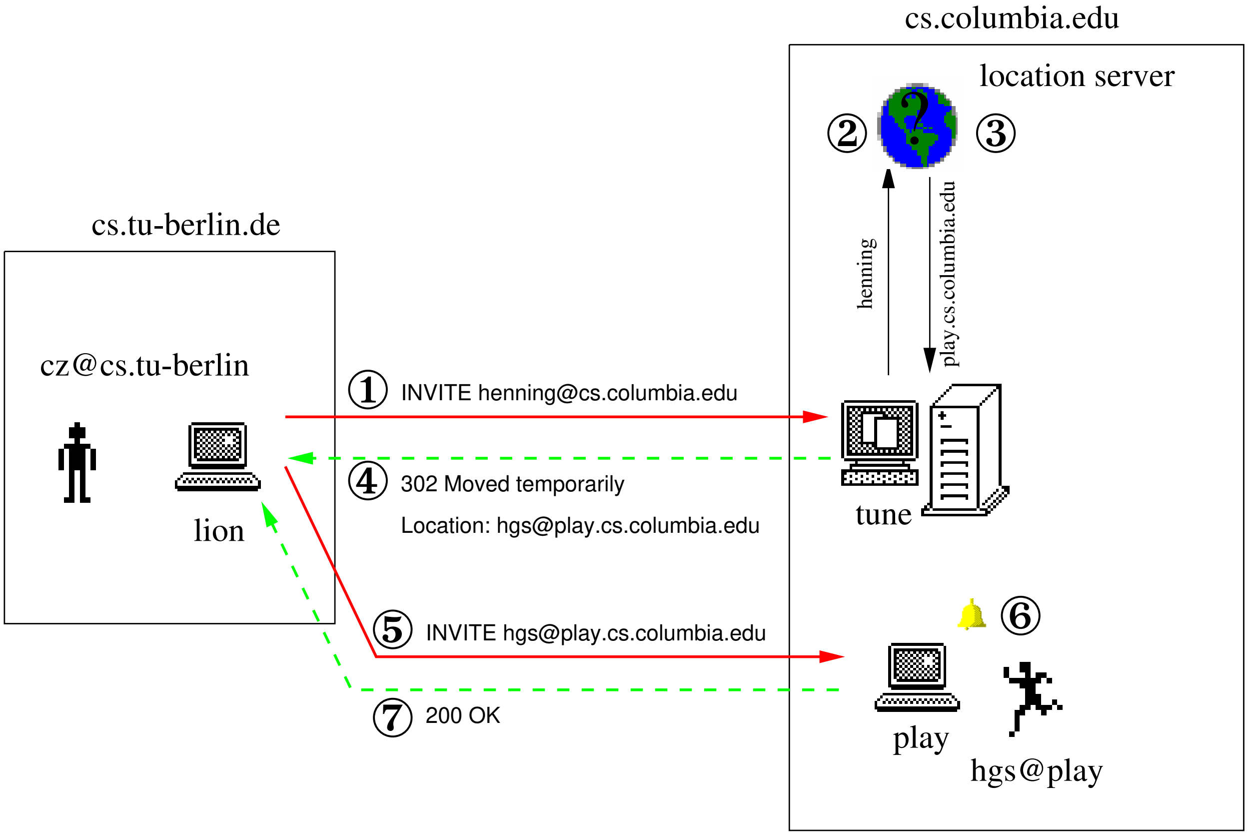

Figure 2: Example of SIP redirect server

use other protocols, such as finger [14], rwho, multicast-based protocols or operating-system dependent mechanism to actively determine the end system where a user is reachable. The location services yield a list of a zero or more possible locations, possibly even sorted in order of likelihood of success.

The location server can be part of the SIP server or the SIP server may use a different protocol (e.g., finger [14] or LDAP [15]) to map addresses. A single user may be registered at different locations, either because she is logged in at several hosts simultaneously or because the location server has (temporarily) inaccurate information.

The action taken on receiving a list of locations varies with the type of SIP server. A SIP redirect server simply returns the list to the client sending the request as Location headers (Section 6.17). A SIP proxy server can sequentially try the addresses until the call is successful (2xx response) or the callee has declined the call (40x response). Alternatively, the server may issue several requests in parallel. A proxy server can only issue more than one sequential or parallel connection request if it is the first in the chain of hosts noted in the Via header to do so. If it is not the first, it must issue a redirect response.

If a proxy server forwards a SIP request, it SHOULD add itself to the end of the list of forwarders noted in the Via (Section 6.31) headers. A proxy server also notes whether it is attempting to reach several possible locations at once ("connection forking"). The Via trace ensures that replies can take the same path back, thus ensuring correct operation through compliant firewalls and loop-free requests. On the reply path, each host most remove its Via, so that routing internal information is hidden from the callee and outside networks. When a multicast request is made, first the host making the request, then the multicast address itself are added to the path.

As discussed in Section 1.6, a SIP address may designate a group rather than an individual. A client indicates using the Reach request header whether it wants to reach the first available individual or treat the address as a group, to be invited as a whole. The default is to attempt to reach the first available callee. If the address is designated as a group address, a proxy server MUST return the list of individuals instead of attempting to connect to these.

Otherwise, the proxy cannot report errors and call status appropriately.

2 SIP Uniform Resource Locators

SIP URLs are used within SIP messages to indicate the originator and recipient of a SIP request, and to specify redirection addresses. A SIP URL may be embedded in web pages or other hyperlinks to indicate that a user or service may be called. Within SIP messages, an email address could have been used, but this would have made it more difficult to gateway between SIP and other protocols with other addressing schemes.

For greater functionality, because interaction with some resources may require message headers or message bodies to be specified as well as the SIP address, the sip URL scheme is extended to allow setting SIP request-header fields and the SIP message-body.

A SIP URL follows the guidelines of RFC 1630 [16, 17] and takes the following form:

```
SIP-URL          =  short-sip-url | full-sip-url
full-sip-url     =  ”sip://” user [ ”:” password ] ”@” host
                    url-parameters [ headers ]
short-sip-url    =  user [ ”:” password ] ”@” host : port
user             =  ; defined in RFC 1738 [18]
host             =  ; defined in RFC 1738
port             =  *digit
url-parameters   =  *( ”;” url-parameter)
url-parameter    =  transport-param |
                    ttl-param | maddr-param
transport-param  =  ”transport=” ( ”udp” | ”tcp” )
ttl-param        =  ”ttl=” ttl
ttl              =  1*3DIGIT                             ; 0 to 255
maddr-param      =  ”maddr=” maddr
maddr            =  ; dotted decimal multicast address
headers          =  ”?” header *( ”                      ” header )
header           =  hname ”=” hvalue
hname            =  *urlc
hvalue           =  *urlc
urlc             =  ; defined in [17]
```

Thus a SIP URL can take either a short form or a full form. The short form MAY only be used within SIP messages where the scheme (SIP) can be assumed. In all other cases, and when parameters are required to be specified, the full form MUST be used.

Note that all URL reserved characters must be encoded. The special hname ”body” indicates that the associated hvalue is the message-body of the SIP INVITE request. Within sip URLs, the characters ”?”, ”=”, ”&” are reserved.

Examples of short and full form SIP URLs with identical address are:

```
j.doe@big.com
sip://j.doe@big.com
sip://j.doe:secret@big.com;transport=tcp
sip://j.doe@big.com?subject=project%20x&priority=urgent
```

The password parameter allows to easily specify a call-back address on a secure web page, but carries the same security risks as all URL-based passwords and should only be used under special circumstances where security requirements are low or all transport paths are secured.

Within a SIP message, URLs are used to indicate the source and intended destination of a request, redirection addresses and the current destination of a request. Normally all these fields will contain SIP URLs. When additional parameters are not required, the short form SIP URL can be used unambiguously.

In some circumstances a non-SIP URL may be used in a SIP message. An example might be making a call from a telephone which is relayed by a gateway onto the internet as a SIP request. In such a case, the source of the call is really the telephone number of the caller, and so a SIP URL is inappropriate and a phone URL might be used instead. Thus where SIP specifies user addresses it allows these addresses to be URLs.

Clearly not all URLs are appropriate to be used in a SIP message as a user address. It is unlikely, for example, that HTTP or FTP URLs are useful in this context. The correct behavior when an unknown scheme is encountered by a SIP server is defined in the context of each of the header fields that use a SIP URL.

SIP URLs can define specific parameters of the request, including the transport mechanism (UDP or TCP) and the use of multicast to make a request. These parameters are added after the host and are separated by semi-colons. For example, to specify to call `j.doe@big.com` using multicast to 239.255.255.1 with a ttl of 15, the following URL would be used:

```
sip://j.doe@big.com;maddr=239.255.255.1;ttl=15
```

The transport protocol UDP is to be assumed when a multicast address is given.

3 SIP Message Overview

Since much of the message syntax is identical to HTTP/1.1, rather than repeating it here we use [HX.Y] to refer to Section X.Y of the current HTTP/1.1 specification [9]. In addition, we describe SIP in both prose and an augmented Backus-Naur form (BNF) [H2.1] described in detail in [19].

All SIP messages are text-based and use HTTP/1.1 conventions [H4.1], except for the additional ability of SIP to use UDP. When sent over TCP or UDP, multiple SIP transactions can be carried in a single TCP connection or UDP datagram. UDP datagrams, including all headers, should not normally be larger than the path maximum transmission unit (MTU) if the MTU is known, or 1500 bytes if the MTU is unknown.

> The 1400 bytes accommodates lower-layer packet headers within the "typical" MTU of around 1500 bytes. There are few MTU values around 1 kB; the next value is 1006 bytes for SLIP and 296 for low-delay PPP [20]. Recent studies [21, p. 154] indicate that an MTU of 1500 bytes is a reasonable assumption. Thus, another reasonable value would be a message size of 950 bytes, to accommodate packet headers within the SLIP MTU without fragmentation.

A SIP message is either a request from a client to a server, or a response from a server to a client.

SIP-message = Request | Response ; SIP messages

Both Request (section 4) and Response (section 5) messages use the generic message format of RFC 822 [22] for transferring entities (the payload of the message). Both types of message consist of a start-line, one or more header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the carriage-return line-feed (CRLF)) indicating the end of the header fields, and an optional message-body. To avoid confusion with similar-named headers in HTTP, we refer to the header describing the message body as entity headers. These components are described in detail in the upcoming sections.

```
generic-message  =  start-line
                    *message-header
                    CRLF
                    [ message-body ]

start-line       =  Request-Line | Status-Line


Request   =  Request-Line          ; Section 4.1
             *( general-header
             | request-header
             | entity-header )
             CRLF
             [ message-body ]

Response  =  Status-Line           ; Section 5.1
             *( general-header
             | response-header
             | entity-header )
             CRLF
             [ message-body ]
```

In the interest of robustness, any leading empty line(s) MUST be ignored. In other words, if the Request or Response message begins with a CRLF, the CRLF should be ignored.

4 Request

The Request message format is shown below:

```
Request  =  Request-Line        ; Section 4.1
            *( general-header
            | request-header
            | entity-header )
            CRLF
            [ message-body ]    ; Section 8
```

general-header	=	Call-ID	; Section 6.11
	|	Date	; Section 6.14
	|	Expires	; Section 6.15
	|	From	; Section 6.16
	|	Sequence	; Section 6.26
	|	Via	; Section 6.31
entity-header	=	Content-Length	; Section 6.12
	|	Content-Type	; Section 6.13
request-header	=	Accept	; Section 6.6
	|	Accept-Language	; Section 6.7
	|	Authorization	; Section 6.9
	|	Organization	; Section 6.18
	|	Priority	; Section 6.20
	|	Proxy-Authorization	; Section 6.22
	|	Reach	; Section 6.24
	|	Subject	; Section 6.28
	|	To	; Section 6.29
	|	User-Agent	; Section 6.30
response-header	=	Location	; Section 6.17
	|	Proxy-Authenticate	; Section 6.21
	|	Public	; Section 6.23
	|	Retry-After	; Section 6.25
	|	Server	; Section 6.27
	|	Warning	; Section 6.32
	|	WWW-Authenticate	; Section 6.33

Table 1: SIP headers

4.1 Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF are allowed except in the final CRLF sequence.

Request-Line = Method SP Request-URI SP SIP-Version CRLF

4.1.1 Methods

The following methods are defined:

```
method  =  "INVITE" | "CONNECTED" | "OPTIONS" | "BYE"
        |  "REGISTER" | "UNREGISTER"
```

INVITE: The user or service is being invited to participate in the session. This method MUST be supported by a SIP server.

CONNECTED: A CONNECTED request confirms that the client has received a successful response to an INVITE request. See Section 11 for details. This method MUST be supported by a SIP server.

OPTIONS: The client is being queried as to its capabilities. A server that believes it can contact the user, such as a user agent where the user is logged in and has been recently active, MAY respond to this request with a capability set. Support of this method is OPTIONAL.

BYE: The client indicates to the server that it wishes to abort the call attempt. The leaving party can use a Location header field to indicate that the recipient of request should contact the named address. This implements the "call transfer" telephony functionality. A client SHOULD also use this method to indicate to the callee that it wishes to abort an on-going call attempt.

With UDP, the caller has no other way to signal her intent to drop the call attempt and the callee side will keep "ringing".

When using TCP, a client MAY also close the connection to abort a call attempt. Support of this method is OPTIONAL.

REGISTER: A client uses the REGISTER method to register the address listed in the request line to a SIP server. In the future, the server MAY use the source address and port to forward SIP requests to. A server SHOULD silently drop the registration after one hour, unless refreshed by the client. A server may set or lower or higher refresh interval and indicate the interval through the Expires header (Section 6.15). A single address (if host-independent) may be registered from several different clients. Support of this method is OPTIONAL.

Beyond its use as a simple location service, this method is needed if there are several SIP servers on a single host, so that some cannot use the default port number. Each such server would register with a server for the administrative domain.

UNREGISTER: A client cancels an existing registration established for the Request-URI with REGISTER with the UNREGISTER method. If it unregisters a Request-URI unknown to the servers, the server returns a 200 (OK) response. Support of this method is OPTIONAL.

BYE and REGISTER are experimental and need to be discussed.

Methods that are not supported by a proxy server SHOULD be treated by that proxy as if they were an INVITE method, and relayed through unchanged or cause a redirection as appropriate.

Methods that are not supported by a server should cause a "501 Not Implemented" response to be returned (Section 7).

4.1.2 Request-URI

The Request-URI field is a SIP URL as described in Section 2 or a general URI. It indicates the user or service that this request is being addressed to. Unlike the To field, the Request-URI field may be re-written by proxies. For example, a proxy may perform a lookup on the contents of the To field to resolve a username from a mail alias, and then use this username as part of the Request-URI field of requests it generates.

If a SIP server receives a request contain a URI indicating a scheme other than SIP which that server does not understand, the server MUST return a "400 Bad Request" response. It MUST do this even if the To field contains a scheme it does understand.

4.1.3 SIP Version

Both request and response messages include the version of SIP in use, and basically follow [H3.1], with HTTP replaced by SIP. To be conditionally compliant with this specification, applications sending SIP messages MUST include a SIP-Version of "SIP/2.0".

5 Response

After receiving and interpreting a request message, the recipient responds with a SIP response message. The response message format is shown below:

```
Response  =  Status-Line            ; Section 5.1
             *( general-header
             | response-header
             | entity-header )
             CRLF
             [ message-body ]       ; Section 8
```

[H6] applies except that HTTP-Version is replaced by SIP-Version. Also, SIP defines additional response codes and does not use some HTTP codes.

5.1 Status-Line

The first line of a Response message is the Status-Line, consisting of the protocol version ((Section 4.1.3) followed by a numeric Status-Code and its associated textual phrase, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

```
Status-Line = SIP-version SP Status-Code SP Reason-Phrase CRLF
```

5.1.1 Status Codes and Reason Phrases

The Status-Code is a 3-digit integer result code that indicates the outcome of the attempt to understand and satisfy the request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata, whereas the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason-Phrase.

We provide an overview of the Status-Code below, and provide full definitions in section 7. The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. SIP/2.0 allows 6 values for the first digit:

1xx: Informational – request received, continuing process;

2xx: Success – the action was successfully received, understood, and accepted;

3xx: Redirection – further action must be taken in order to complete the request;

4xx: Client Error – the request contains bad syntax or cannot be fulfilled at this server;

5xx: Server Error – the server failed to fulfill an apparently valid request;

6xx: Global Failure - the request is invalid at any server.

Presented below are the individual values of the numeric response codes, and an example set of corresponding reason phrases for SIP/2.0. These reason phrases are only recommended; they may be replaced by local equivalents without affecting the protocol. Note that SIP adopts many HTTP/1.1 response codes. SIP/2.0 adds response codes in the range starting at x80 to avoid conflicts with newly defined HTTP response codes, and extends these response codes in the 6xx range.

SIP response codes are extensible. SIP applications are not required to understand the meaning of all registered response codes, though such understanding is obviously desirable. However, applications MUST understand the class of any response code, as indicated by the first digit, and treat any unrecognized response as being equivalent to the x00 response code of that class, with the exception that an unrecognized response MUST NOT be cached. For example, if a client receives an unrecognized response code of 431, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 response code. In such cases, user agents SHOULD present to the user the message body returned with the response, since that message body is likely to include human-readable information which will explain the unusual status.

6 Header Field Definitions

SIP header fields are similar to HTTP header fields in both syntax and semantics [H4.2], [H14]. In general the ordering of the header fields is not of importance (with the exception of Via fields, see below), but proxies MUST NOT reorder or otherwise modify header fields other than by adding a new Via field. This allows an authentication field to be added after the Via fields that will not be invalidated by proxies.

To, From, and Call-ID header MUST be present in each request with method INVITE. The Content-Type and Content-Length headers are required when there is a valid message body (of non-zero length) associated with the message (Section 8).

A server MUST understand the PEP-Require header.

Other headers may be added as required; a server MAY ignore headers that it does not understand. A compact form of these header fields is also defined in Section 10 for use over UDP when the request has to fit into a single packet and size is an issue.

6.1 General Header Fields

There are a few header fields that have general applicability for both request and response messages. These header fields apply only to the message being transmitted.

General-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of general header fields if all parties in the communication recognize them to be general-header fields.

```
Status-Code     =  "100"          ; Trying
                |  "180"          ; Ringing
                |  "200"          ; OK
                |  "300"          ; Multiple Choices
                |  "301"          ; Moved Permanently
                |  "302"          ; Moved Temporarily
                |  "303"          ; See Other
                |  "305"          ; Use Proxy
                |  "380"          ; Alternative Service
                |  "400"          ; Bad Request
                |  "401"          ; Unauthorized
                |  "402"          ; Payment Required
                |  "403"          ; Forbidden
                |  "404"          ; Not Found
                |  "405"          ; Method Not Allowed
                |  "407"          ; Proxy Authentication Required
                |  "408"          ; Request Timeout
                |  "409"          ; Conflict
                |  "410"          ; Gone
                |  "411"          ; Length Required
                |  "412"          ; Precondition Failed
                |  "413"          ; Request Message Body Too Large
                |  "414"          ; Request-URI Too Large
                |  "415"          ; Unsupported Media Type
                |  "420"          ; Bad Extension
                |  "480"          ; Temporarily not available
                |  "500"          ; Internal Server Error
                |  "501"          ; Not Implemented
                |  "502"          ; Bad Gateway
                |  "503"          ; Service Unavailable
                |  "504"          ; Gateway Timeout
                |  "505"          ; SIP Version not supported
                |  "600"          ; Busy
                |  "603"          ; Decline
                |  "604"          ; Does not exist anywhere
                |  "606"          ; Not Acceptable
                |  extension-code
extension-code  =  3DIGIT

Reason-Phrase   =  *<TEXT, excluding CR, LF>
```

Figure 3: Status Codes

6.2 Entity Header Fields

Entity-header fields define meta-information about the message-body or, if no body is present, about the resource identified by the request. The term "entity header" is an HTTP 1.1 term where the reply body may contain a transformed version of the message body. The original message body is referred to as the "entity". We retain the same terminology for header fields but usually refer to the "message body" rather then the entity as the two are the same in SIP.

6.3 Request Header Fields

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation.

Request-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields MAY be given the semantics of request-header fields if all parties in the communication recognize them to be request-header fields. Unrecognized header fields are treated as entity-header fields.

6.4 Response Header Fields

The response-header fields allow the server to pass additional information about the response which cannot be placed in the Status- Line. These header fields give information about the server and about further access to the resource identified by the Request-URI.

Response-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields MAY be given the semantics of response-header fields if all parties in the communication recognize them to be response-header fields. Unrecognized header fields are treated as entity-header fields.

6.5 Header Field Format

Header fields (general-header, request-header, response-header, and entity-header) follow the same generic header format as that given in Section 3.1 of RFC 822 [22].

Each header field consists of a name followed by a colon (":") and the field value. Field names are case-insensitive. The field value may be preceded by any amount of leading white space (LWS), though a single space (SP) is preferred. Header fields can be extended over multiple lines by preceding each extra line with at least one SP or horizontal tab (HT). Applications SHOULD follow HTTP "common form" when generating these constructs, since there might exist some implementations that fail to accept anything beyond the common forms.

```
message-header  =  field-name ":" [ field-value ] CRLF
field-name      =  token
field-value     =  *( field-content — LWS )
field-content   =  <the OCTETs making up the field-value
                   and consisting of either *TEXT or combinations
                   of token, tspecials, and quoted-string>
```

The order in which header fields are received is not significant if the header fields have different field names. Multiple header fields with the same field-name may be present in a message if and only if the entire field-value for that header field is defined as a comma-separated list (i.e., `#(values)`). It MUST be possible to combine the multiple header fields into one "field-name: field-value" pair, without changing the semantics of the message, by appending each subsequent field-value to the first, each separated by a comma. The order in which header fields with the same field-name are received is therefore significant to the interpretation of the combined field value, and thus a proxy MUST NOT change the order of these field values when a message is forwarded.

Field names are not case-sensitive, although their values may be.

6.6 Accept

See [H14.1]. This request header field is used only with the OPTIONS request to indicate what description formats are acceptable.

Example:

```
Accept: application/sdp;level=1, application/x-private
```

6.7 Accept-Language

See [H14.4]. The Accept-Language request header can be used to allow the client to indicate to the server in which language it would prefer to receive reason phrases. This may also be used as a hint by the proxy as to which destination to connect the call to (e.g., for selecting a human operator).

Example:

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```

6.8 Allow

See [H14.7].

6.9 Authorization

See [H14.8].

6.10 Authentication

Authentication fields provide a digital signature of the remaining fields for authentication purposes. They are *not yet defined*. The use of authentication headers is optional. If used, authentication headers MUST be added to the header after the Via fields and before the rest of the fields.

> HS: Should probably re-use S/MIME here rather than invent our own. Maybe better to fold into Authorization field.

6.11 Call-ID

The Call-ID uniquely identifies a particular invitation. Note that a single multimedia conference may give rise to several calls, e.g., if a user invites several different people. Calls to different callee MUST always use different Call-IDs unless they are the result of a proxy server "forking" a single request.

The Call-ID may be any URL-encoded string that can be guaranteed to be globally unique for the duration of the request. Using the initiator's IP-address, process id, and instance (if more than one request is being made simultaneously) satisfies this requirement.

The form local-id@host is recommended, where host is either the fully qualified domain name or a globally routable IP address, and local-id depends on the application and operating system of the host, but is an ID that can be guaranteed to be unique during this session initiation request.

```
Call-ID  =  ( "Call-ID" | "i" ) ":" atom "@" host
```

Example:

```
Call-ID: 9707211351.AA08181@foo.bar.com
```

6.12 Content-Length

The Content-Length entity-header field indicates the size of the message-body, in decimal number of octets, sent to the recipient.

```
Content-Length = "Content-Length" ":" 1*DIGIT
```

An example is

```
Content-Length: 3495
```

Applications SHOULD use this field to indicate the size of the message-body to be transferred, regardless of the media type of the entity. Any Content-Length greater than or equal to zero is a valid value. If no body is present in a message, then the Content-Length header MAY be omitted or set to zero. Section 8 describes how to determine the length of the message body.

6.13 Content-Type

The Content-Type entity-header field indicates the media type of the message-body sent to the recipient.

Content-Type = "Content-Type" ":" media-type

An example of the field is

```
Content-Type: application/sdp
```

6.14 Date

See [H14.19].

The Date header field is useful for simple devices without their own clock.

6.15 Expires

The Expires header field gives the date/time after which the registration expires.

This header field is currently defined only for the REGISTER and INVITE methods. For REGISTER, it is a response-header field and allows the server to indicate when the client has to re-register. For INVITE, it is a request-header with which the callee can limit the validity of an invitation. (For example, if a client wants to limit how long a search should take at most or when a conference being invited to is time-limited. A user interface may take this is as a hint to leave the invitation window on the screen even if the user is not currently at the workstation.)

The value of this field can be either an HTTP-date or an integer number of seconds (in decimal), measured from the receipt of the request.

Expires = ”Expires” ”:” (HTTP-date | delta-seconds)

Two example of its use are

```
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Expires: 5
```

6.16 From

Requests MUST and responses SHOULD contain a From header field, indicating the invitation initiator. The field MUST be a SIP URL as defined in Section 2. Only a single initiator and a single invited user are allowed to be specified in a single SIP request. The sense of To and From header fields is maintained from request to response, i.e., if the From header is `sip://bob@example.edu` in the request then it is MUST also be `sip://bob@example.edu` in the response to that request.

The From field is a URL and not a simple SIP address (Section 1.6 address to allow a gateway to relay a call into a SIP request and still produce an appropriate From field. An example might be a telephone call relayed into a SIP request where the from field might contain a phone:// URL. Normally however this field will contain a sip:// URL in either the long or short form.

If a SIP agent or proxy receives a request sourced From a URL indicating a scheme other that SIP that is unknown to it, this MUST NOT be treated as an error.

From = (”From” | ”f”) ”:” *1((SIP-URL | URL) [comment])

Example:

```
From: mjh@isi.edu (Mark Handley)
```

6.17 Location

The Location response header can be used with a 2xx or 3xx response codes to indicate a new location to try. It contains a SIP URL giving the new location or username to try, or may simply specify addition transport parameters. For example, a “301 Moved Permanently” response SHOULD contain a Location field containing the SIP URL giving the new location and username to try. However, a “302 Moved Temporarily” MAY give simply the same location and username that was being tried but specify additional transport parameters such as a multicast address to try or a change of transport from UDP to TCP or vice versa.

A user agent or redirect server sending a definitive, positive response (2xx), SHOULD insert a Location response header indicating the SIP address under which it is reachable most directly for future SIP requests. This may be the address of the server itself or that of a proxy (e.g., if the host is behind a firewall).

```
Location             =  ( ”Location” | ”m” ) ( SIP-URL | URL )
                        *( ”;” location-params )
extension-name       =  token
extension-value      =  *( token | quoted-string | LWS | extension-specials)
extension-specials   =  < any element of tspecials except <”> >
language-tag         =  < see [H3.10] >
service-tag          =  ”fax” | ”IP” | ”PSTN” | ”ISDN” | ”pager” | ”voice-mail
                        | ”attendant”
media-tag            =  < see SDP: ”audio” | ”video” | . . .
feature-list         =  to be determined
location-params      =  ”q”                  ”=”   qvalue
                     |  ”mobility”           ”=”   ( ”fixed” | ”mobile” )
                     |  ”class”              ”=”   ( ”personal” | ”business” )
                     |  ”language”           ”=”   1# language-tag
                     |  ”service”            ”=”   1# service-tag
                     |  ”media”              ”=”   1# media-tag
                     |  ”features”           ”=”   1# feature-list
                     |  ”description”        ”=”   quoted-string
                     |  ”duplex”             ”=”   ( ”full” | ”half” | ”receive-only” |
                                                   ”send-only” )
                     |  extension-attributes
extension-attribute  =  extension-name       ”=”   extension-value
```

Examples:

```
Location: sip://hgs@erlang.cs.columbia.edu ;service=IP,voice-mail
          ;media=audio ;duplex=full ;q=0.7
Location: phone://1-415-555-1212 ; service=ISDN;mobility=fixed;
          language=en,es,iw ;q=0.5
Location: phone://1-800-555-1212 ; service=pager;mobility=mobile;
          duplex=send-only;media=text; q=0.1
```

Attributes which are unknown should be omitted. New tags for class-tag and service-tag can be registered with IANA. The media tag uses Internet media types, e.g., audio, video, application/x-wb, etc. This is meant for indicating general communication capability, sufficient for the caller to choose an appropriate address.

6.18 Organization

The Organization request-header fields conveys the name of the organization to which the callee belongs. It may be inserted by proxies at the boundary of an organization and may be used by client software to filter calls.

6.19 PEP

This corresponds to the PEP header in the "Protocol Extension Protocol" defined in RFC XXXX. The Protocol Extension Protocol (PEP) is an extension mechanism designed to accommodate dynamic extension of applications such as SIP clients and servers by software components. The PEP general header declares new headers and whether an application must or may understand them. Servers MUST parse this field and MUST return "420 Bad Extension" when there is a PEP extension of strength "must" (see RFC XXXX) that they do not understand.

6.20 Priority

The priority request header signals the urgency of the call to the callee.

Priority	=	"Priority" ":" priority-value
priority-value	=	"urgent" | "normal" | "non-urgent"

Example:

```
Subject: A tornado is heading our way!
Priority: urgent
```

6.21 Proxy-Authenticate

See [H14.33].

6.22 Proxy-Authorization

See [H14.34].

6.23 Public

See [H14.35].

6.24 Reach

The Reach request header field allows the client to indicate whether it wants to reach the group identified by the *user* part of the address (value "all") or the first available individual (value "first"). If not present, a value of “first” is implied. The “do-not-forward” request prohibits proxies from forwarding the call to another individual (e.g., the call is personal or the caller does not want to be shunted to a secretary if the line is busy.) Section 1.6 describes the behavior of proxy servers when resolving group aliases.

Reach = ”Reach” ”:” 1#(”first” | ”all”) (”do-not-forward”)

Example:

```
Reach: first, do-not-forward
```

HS: This header is experimental.

6.25 Retry-After

The Retry-After response header field can be used with a “503 Service Unavailable” response to indicate how long the service is expected to be unavailable to the requesting client and with a “404 Not Found” or “451 Busy” response to indicate when the called party may be available again. The value of this field can be either an HTTP-date or an integer number of seconds (in decimal) after the time of the response.

Retry-After = ”Retry-After” ”:” (HTTP-date | delta-seconds)

Two examples of its use are

```
Retry-After: Mon, 21 Jul 1997 18:48:34 GMT
Retry-After: 120
```

In the latter example, the delay is 2 minutes.

6.26 Sequence

The Sequence header field MAY be added by a SIP client making a request if it needs to distinguish responses to several consecutive requests sent with the same Call-ID. A Sequence field contains a single decimal sequence number chosen by the requesting client. Consecutive different requests made with the same Call-ID MUST contain strictly monotonically increasing sequence numbers although the sequence space MAY NOT be contiguous. A server responding to a request containing a sequence number MUST echo the sequence number back in the response.

Sequence = ”Sequence” ”:” 1*DIGIT

Sequence header fields are NOT needed for SIP requests using the INVITE or OPTIONS methods but may be needed for future methods.

Example:

```
Sequence: 4711
```

6.27 Server

See [H14.39].

6.28 Subject

This is intended to provide a summary, or indicate the nature, of the call, allowing call filtering without having to parse the session description. (Also, the session description may not necessarily use the same subject indication as the invitation.)

Subject = ("Subject" | "s") ":" *text

Example:

```
Subject: Tune in - they are talking about your work!
```

6.29 To

The To request header field specifies the invited user, with the same SIP URL syntax as the From field.

To = ("To" | "t") ":" (SIP-URL | URL) [comment]

If a SIP server receives a request destined To a URL indicating a scheme other than SIP and that is unknown to it, the server returns a "400 bad request" response.

Example:

```
To: sip://operator@cs.columbia.edu (The Operator)
```

6.30 User-Agent

See [H14.42].

6.31 Via

The Via field indicates the path taken by the request so far. This prevents request looping and ensures replies take the same path as the requests, which assists in firewall traversal and other unusual routing situations.

In the request path, an initiator MUST add its own Via field to each request. This Via field MUST be the first field in the request. Each subsequent client or proxy that sends the message onwards MUST add its own additional Via field, which MUST be added before any existing Via fields. Additionally, if the message goes to a multicast address, an extra Via field is added before all the others giving the multicast address and TTL.

In the return path, Via fields are processed by a proxy or client according to the following rules:

- If the first Via field in the reply **received** is the client's or server's local address, remove the Via field and process the reply.
- If the first Via field in a reply you are going to **send** is a multicast address, remove that Via field **before** sending to the multicast address.

These rules ensure that a client or proxy server only has to check the first Via field in a reply to see if it needs processing.

When a reply passes through a proxy on the reverse path, that proxies Via field MUST be removed from the reply.

The format for a Via header is:

```
Via               =  ( "Via" | "v") ":" 1#( sent-protocol sent-by
                     *( ";" via-params ) [ comment ] )
via-params        =  "ttl" "=" ttl
                  |  "fanout"
sent-protocol     =  [ protocol-name "/" ] protocol-version
 [ "/" transport ]
protocol-name     =  "SIP" | token
protocol-version  =  token
transport         =  "UDP" | "TCP"
sent-by           =  host [ ":" port ]
ttl               =  1*3DIGIT                          ; 0 to 255
```

The "ttl" parameter is included only if the address is a multicast address. The "fanout" parameter indicates that this proxy has initiated several connection attempts and that subsequent proxies should not do the same.

Example:

```
Via: SIP/2.0/UDP first.example.com:4000 ;fanout
```

6.32 Warning

The Warning response-header field is used to carry additional information about the status of a response. Warning headers are sent with responses using:

Warning	=	"Warning" ":" 1#warning-value
warning-value	=	warn-code SP warn-agent SP warn-text
warn-code	=	2DIGIT
warn-agent	=	(host [":" port]) | pseudonym
		; the name or pseudonym of the server adding
		; the Warning header, for use in debugging
warn-text	=	quoted-string

A response may carry more than one Warning header.

The warn-text should be in a natural language and character set that is most likely to be intelligible to the human user receiving the response. This decision may be based on any available knowledge, such as the location of the cache or user, the Accept-Language field in a request, the Content-Language field in a response, etc. The default language is English and the default character set is ISO- 8859-1.

Any server may add Warning headers to a response. New Warning headers should be added after any existing Warning headers. A proxy server MUST NOT delete any Warning header that it received with a response.

When multiple Warning headers are attached to a response, the user agent SHOULD display as many of them as possible, in the order that they appear in the response. If it is not possible to display all of the warnings, the user agent should follow these heuristics:

- Warnings that appear early in the response take priority over those appearing later in the response.
- Warnings in the user's preferred character set take priority over warnings in other character sets but with identical warn-codes and warn-agents.

Systems that generate multiple Warning headers should order them with this user agent behavior in mind.

Example:

```
Warning: 606.4 isi.edu Multicast not available
Warning: 606.2 isi.edu Incompatible protocol (RTP/XXP)
```

6.33 WWW-Authenticate

See [H14.46].

7 Status Code Definitions

The response codes are consistent with, and extend, HTTP/1.1 response codes. Not all HTTP/1.1 response codes are appropriate, and only those that are appropriate are given here. Response codes not defined by HTTP/1.1 have codes x80 upwards to avoid clashes with future HTTP response codes. Also, SIP defines a new class, 6xx. The default behavior for unknown response codes is given for each category of codes.

7.1 Informational 1xx

Informational responses indicate that the server or proxy contacted is performing some further action and does not yet have a definitive response. The client SHOULD wait for a further response from the server, and the server SHOULD send such a response without further prompting. If UDP transport is being used, the client SHOULD periodically re-send the request in case the final response is lost. Typically a server should send a “1xx” response if it expects to take more than one second to obtain a final reply.

7.1.1 100 Trying

Some further action is being taken (e.g., the request is being forwarded) but the user has not yet been located.

7.1.2 180 Ringing

The user agent or conference server has located a possible location where the user has been recently and is trying to alert them.

7.2 Successful 2xx

The request was successful and MUST terminate a search.

7.2.1 200 OK

The request was successful in contacting the user, and the user has agreed to participate.

7.3 Redirection 3xx

3xx responses give information about the user’s new location, or about alternative services that may be able to satisfy the call. They SHOULD terminate an existing search, and MAY cause the initiator to begin a new search if appropriate.

7.3.1 300 Multiple Choices

The requested resource corresponds to any one of a set of representations, each with its own specific location, and agent-driven negotiation (i.e., controlled by the SIP client) is being provided so that the user (or user agent) can select a preferred communication end point and redirect its request to that location.

The response SHOULD include an entity containing a list of resource characteristics and location(s) from which the user or user agent can choose the one most appropriate. The entity format is specified by the media type given in the Content-Type header field. Depending upon the format and the capabilities of the user agent, selection of the most appropriate choice may be performed automatically. However, this specification does not define any standard for such automatic selection.

The choices SHOULD also be listed as Location fields (Section 6.17). Unlike HTTP, the SIP response may contain several Location fields. User agents MAY use the Location field value for automatic redirection or MAY ask the user to confirm a choice.

7.3.2 301 Moved Permanently

The requesting client should retry on the new address given by the Location field because the user has permanently moved and the address this response is in reply to is no longer a current address for the user. A 301 response MUST NOT suggest any of the hosts in the Via path of the request as the user's new location.

7.3.3 302 Moved Temporarily

The requesting client should retry on the new address(es) given by the Location header. A 302 response MUST NOT suggest any of the hosts in the Via path of the request as the user's new location.

7.3.4 380 Alternative Service

The call was not successful, but alternative services are possible. The alternative services are described in the message body of the response.

7.4 Request Failure 4xx

4xx responses are definite failure responses from a particular server. The client SHOULD NOT retry the same request without modification (e.g., adding appropriate authorization). However, the same request to a different server may be successful.

7.4.1 400 Bad Request

The request could not be understood due to malformed syntax.

7.4.2 401 Unauthorized

The request requires user authentication.

7.4.3 402 Payment Required

Reserved for future use.

7.4.4 403 Forbidden

The server understood the request, but is refusing to fulfill it. Authorization will not help, and the request should not be repeated.

7.4.5 404 Not Found

The server has definitive information that the user does not exist at the domain specified in the Request-URI.

7.4.6 405 Method Not Allowed

The method specified in the Request-Line is not allowed for the address identified by the Request-URI. The response MUST include an Allow header containing a list of valid methods for the indicated address.

7.4.7 407 Proxy Authentication Required

This code is similar to 401 (Unauthorized), but indicates that the client MUST first authenticate itself with the proxy. The proxy MUST return a Proxy-Authenticate header field (section 6.21) containing a challenge applicable to the proxy for the requested resource. The client MAY repeat the request with a suitable Proxy-Authorization header field (section 6.22). SIP access authentication is explained in section [H11].

This status code should be used for applications where access to the communication channel (e.g., a telephony gateway) rather than the callee herself requires authentication.

7.4.8 408 Request Timeout

The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time.

7.4.9 420 Bad Extension

The server did not understand the protocol extension specified with strength “must”.

7.4.10 480 Temporarily Unavailable

The callee’s end system was contacted successfully but the callee is currently unavailable (e.g., not logged in or logged in in such a manner as to preclude communication with the callee). The response may indicate a better time to call in the Retry-After header. The user may also be available elsewhere (unbeknownst to this host), thus, this response does terminate any searches.

7.5 Server Failure 5xx

5xx responses are failure responses given when a server itself has erred. They are not definitive failures, and SHOULD NOT terminate a search if other possible locations remain untried.

7.5.1 500 Server Internal Error

The server encountered an unexpected condition that prevented it from fulfilling the request.

7.5.2 501 Not implemented

The server does not support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any user.

7.5.3 502 Bad Gateway

The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.

7.5.4 503 Service Unavailable

The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. The implication is that this is a temporary condition which will be alleviated after some delay. If known, the length of the delay may be indicated in a Retry-After header. If no Retry-After is given, the client SHOULD handle the response as it would for a 500 response.

Note: The existence of the 503 status code does not imply that a server must use it when becoming overloaded. Some servers may wish to simply refuse the connection.

7.5.5 504 Gateway Timeout

The server, while acting as a gateway, did not receive a timely response from the upstream server (e.g., a location server) it accessed in attempting to complete the request.

7.6 Global Failures

6xx responses indicate that a server has definitive information about a particular user, not just the particular instance indicated in the Request-URI. All further searches for this user are doomed to failure and pending searches SHOULD be terminated.

7.6.1 600 Busy

The callee's end system was contacted successfully but the callee is busy and does not wish to take the call at this time. The response may indicate a better time to call in the Retry-After header. If the callee does not wish to reveal the reason for declining the call, the callee should use status code 680 instead.

7.6.2 603 Decline

The callee's machine was successfully contacted but the user explicitly does not wish to participate. The response may indicate a better time to call in the Retry-After header.

7.6.3 604 Does not exist anywhere

The server has authoritative information that the user indicated in the To request field does not exist anywhere. Searching for the user elsewhere will not yield any results.

7.6.4 606 Not Acceptable

The user's agent was contacted successfully but some aspects of the session profile (the requested media, bandwidth, or addressing style) were not acceptable.

A "606 Not Acceptable" reply means that the user wishes to communicate, but cannot adequately support the session described. The "604 Not Acceptable" reply MAY contain a list of reasons in a Warning header describing why the session described cannot be supported. These reasons can be one or more of:

606.1 Insufficient Bandwidth: The bandwidth specified in the session description or defined by the media exceeds that known to be available.

606.2 Incompatible Protocol: One or more protocols described in the request are not available.

606.3 Incompatible Format: One or more media formats described in the request is not available.

606.4 Multicast not available: The site where the user is located does not support multicast.

606.5 Unicast not available: The site where the user is located does not support unicast communication (usually due to the presence of a firewall).

Other reasons are likely to be added later. It is hoped that negotiation will not frequently be needed, and when a new user is being invited to join a pre-existing lightweight session, negotiation may not be possible. It is up to the invitation initiator to decide whether or not to act on a "606 Not Acceptable" reply.

8 SIP Message Body

The session description body gives details of the session the user is being invited to join. Its Internet media type MUST be given by the Content-type header field, and the body length in bytes MUST be given by the Content-Length header field. If the body has undergone any encoding (such as compression) then this MUST be indicated by the Content-encoding header field, otherwise Content-encoding MUST be omitted.

If required, the session description can be encrypted using public key cryptography, and then can also carry private session keys for the session. If this is the case, four random bytes are added to the beginning of the session description before encryption and are removed after decryption but before parsing.

8.1 Body Inclusion

For a request message, the presence of a body is signaled by the inclusion of a Content-Length header. A body may be included in a request only when the request method allows one.

For response messages, whether or not a body is included is dependent on both the request method and the response message's response code. All 1xx informational responses MUST NOT include a body. All other responses MAY include a payload, although it may be of zero length.

8.2 Message Body Length

If no body is present in a message, then the Content-Length header MAY be omitted or set to zero. When a body is included, its length in bytes is indicated in the Content-Length header and is determined by one of the following:

1. Any response message which MUST NOT include a body (such as the 1xx responses) is always terminated by the first empty line after the header fields, regardless if any entity-header fields are present.
2. Otherwise, a Content-Length header MUST be present (this requirement differs from HTTP/1.1). Its value in bytes represents the length of the message body.

The "chunked" transfer encoding of HTTP/1.1 MUST NOT be used for SIP.

9 Examples

9.1 Invitation

9.1.1 Request

The example below is a request message en route from initiator to invitee:

```
C->S: INVITE schooler@vlsi.cs.caltech.edu SIP/2.0
      Via: SIP/2.0/UDP 239.128.16.254 16
      Via: SIP/2.0/UDP 131.215.131.131
      Via: SIP/2.0/UDP 128.16.64.19
      From: mjh@isi.edu (Mark Handley)
      Subject: SIP will be discussed, too
      To: schooler@cs.caltech.edu (Eve Schooler)
      Call-ID: 62729-27@oregon.isi.edu
      Content-type: application/sdp
      Content-Length: 187

      v=0
      o=user1 53655765 2353687637 IN IP4 128.3.4.5
      s=Mbone Audio
```

```
       i=Discussion of Mbone Engineering Issues
       e=mbone@somewhere.com
       c=IN IP4 224.2.0.1/127
       t=0 0
       m=audio 3456 RTP/AVP 0
```

The first line above states that this is a SIP version 2.0 request.

The Via fields give the hosts along the path from invitation initiator (the first element of the list) towards the invitee. In the example above, the message was last multicast to the administratively scoped group `239.128.16.254` with a ttl of `16` from the host `131.215.131.131`.

The request header above states that the request was initiated by `mjh@isi.edu`. The Via header indicates that is was initiated from the host `128.16.64.19`. `schooler@cs.caltech.edu` is being invited; the message is currently being routed to `schooler@vlsi.cs.caltech.edu`.

In this case, the session description is using the Session Description Protocol (SDP), as stated in the Content-type header.

The header is terminated by an empty line and is followed by a message body containing the session description.

9.1.2 Reply

The called user agent, directly or indirectly through proxy servers, indicates that it is alerting ("ringing") the called party:

```
S->C: SIP/2.0 180 Ringing
       Via: SIP/2.0/UDP 239.128.16.254 16
       Via: SIP/2.0/UDP 131.215.131.131
       Via: SIP/2.0/UDP 128.16.64.19 1
       From: mjh@isi.edu
       Call-ID: 62729-27@128.16.64.19
       Location: sip://es@jove.cs.caltech.edu
```

A sample reply to the invitation is given below. The first line of the reply states the SIP version number, that it is a "200 OK" reply, which means the request was successful. The Via headers are taken from the request, and entries are removed hop by hop as the reply retraces the path of the request. A new authentication field MAY be added by the invited user's agent if required. The Call-ID is taken directly from the original request, along with the remaining fields of the request message. The original sense of From field is preserved (i.e., it is the session initiator).

In addition, the Location header gives details of the host where the user was located, or alternatively the relevant proxy contact point which should be reachable from the caller's host.

```
S->C: SIP/2.0 200 OK
      Via: SIP/2.0/UDP 239.128.16.254 16
      Via: SIP/2.0/UDP 131.215.131.131
      Via: SIP/2.0/UDP 128.16.64.19 1
      From: mjh@isi.edu
      To: schooler@cs.caltech.edu
      Call-ID: 62729-27@128.16.64.19
      Location: sip://es@jove.cs.caltech.edu
```

For two-party Internet phone calls, the response must contain a description of where to send data to, for example the reply from `schooler` to `mjh`:

```
S->C: SIP/2.0 200 OK
      Via: SIP/2.0/UDP 239.128.16.254 16
      Via: SIP/2.0/UDP 131.215.131.131
      Via: SIP/2.0/UDP 128.16.64.19 1
      From: mjh@isi.edu
      To: schooler@cs.caltech.edu
      Call-ID: 62729-27@128.16.64.19
      Location: sip://es@jove.cs.caltech.edu
      Content-Length: 102

      v=0
      o=schooler 4858949 4858949 IN IP4 192.1.2.3
      t=0 0
      m=audio 5004 RTP/AVP 0
      c=IN IP4 131.215.131.147
```

The caller confirms the invitation by sending a request to the location named in the Location header:

```
C->S: CONNECTED schooler@jove.cs.caltech.edu SIP/2.0
      From: mjh@isi.edu
      To: schooler@cs.caltech.edu
      Call-ID: 62729-27@128.16.64.19
```

9.1.3 Aborting a Call

If the caller wants to abort a pending call, it sends a BYE request.

```
C->S: BYE schooler@jove.cs.caltech.edu
      From: mjh@isi.edu
      To: schooler@cs.caltech.edu
      Call-ID: 62729-27@128.16.64.19
```

9.1.4 Redirects

Replies with response codes “301 Moved Permanently” or “302 Moved Temporarily” SHOULD specify another location using the Location field.

```
S->C: SIP/2.0 302 Moved temporarily
      Via: SIP/2.0/UDP 131.215.131.131
      Via: SIP/2.0/UDP 128.16.64.19
      From: mjh@isi.edu
      To: schooler@cs.caltech.edu
      Call-ID: 62729-27@128.16.64.19
      Location: sip://239.128.16.254;ttl=16;transport=udp
      Content-length: 0
```

In this example, the proxy located at `131.215.131.131` is being advised to contact the multicast group `239.128.16.254` with a ttl of `16` and UDP transport. In normal situations, a server would not suggest a redirect to a local multicast group unless, as in the above situation, it knows that the previous proxy or client is within the scope of the local group. If the request is redirected to a multicast group, a proxy server SHOULD query the multicast address itself rather than sending the redirect back towards the client as multicast may be scoped; this allows a proxy within the appropriate scope regions to make the query.

9.1.5 Alternative Services

An example of a “350 Alternative Service” reply is:

```
S->C: SIP/2.0 350 Alternative Service
      Via: SIP/2.0/UDP 131.215.131.131
      Via: SIP/2.0/UDP 128.16.64.19
      From: mjh@isi.edu
      To: schooler@cs.caltech.edu
      Call-ID: 62729-27@128.16.64.19
      Location: recorder@131.215.131.131
      Content-type: application/sdp
      Content-length: 146

      v=0
```

```
       o=mm-server 2523535 0 IN IP4 131.215.131.131
       s=Answering Machine
       i=Leave an audio message
       c=IN IP4 131.215.131.131
       t=0 0
       m=audio 12345 RTP/AVP 0
```

In this case, the answering server provides a session description that describes an "answering machine". If the invitation initiator decides to take advantage of this service, it should send an invitation request to the answering machine at `131.215.131.131` with the session description provided (modified as appropriate for a unicast session to contain the appropriate local address and port for the invitation initiator). This request SHOULD contain a different Call-ID from the one in the original request. An example would be:

```
C->S: INVITE mm-server@131.215.131.131 SIP/2.0
       Via: SIP/2.0/UDP 128.16.64.19
       From: mjh@isi.edu
       To: schooler@cs.caltech.edu
       Call-ID: 62729-28@128.16.64.19
       Content-type: application/sdp
       Content-length: 146

       v=0
       o=mm-server 2523535 0 IN IP4 131.215.131.131
       s=Answering Machine
       i=Leave an audio message
       c=IN IP4 128.16.64.19
       t=0 0
       m=audio 26472 RTP/AVP 0
```

Invitation initiators MAY choose to treat a "350 Alternative Service" reply as a failure if they wish to do so.

9.1.6 Negotiation

An example of a "606 Not Acceptable" reply is:

```
S->C: SIP/2.0 606 Not Acceptable
       From: mjh@isi.edu
       To: schooler@cs.caltech.edu
       Call-ID:62729-27@128.16.64.19
       Location: mjh@131.215.131.131
       Warning: 606.1 Insufficient bandwidth (only have ISDN),
         606.3 Incompatible format,
```

```
         606.4 Multicast not available
       Content-Type: application/sdp
       Content-Length: 50

       v=0
       s=Lets talk
       b=CT:128
       c=IN IP4 131.215.131.131
       m=audio 3456 RTP/AVP 7 0 13
       m=video 2232 RTP/AVP 31
```

In this example, the original request specified 256 kb/s total bandwidth, and the reply states that only 128 kb/s is available. The original request specified GSM audio, H.261 video, and WB whiteboard. The audio coding and whiteboard are not available, but the reply states that DVI, PCM or LPC audio could be supported in order of preference. The reply also states that multicast is not available. In such a case, it might be appropriate to set up a transcoding gateway and re-invite the user.

9.2 OPTIONS Request

A caller Alice can use an OPTIONS request to find out the capabilities of a potential callee Bob, without "ringing" the designated address. In this case, Bob indicates that he can be reached at three different addresses, ranging from voice-over-IP to a PSTN phone to a pager.

```
C->S: OPTIONS bob@example.com SIP/2.0
      From: alice@anywhere.org (Alice)
      To: bob@example.com (Bob)
      Accept: application/sdp

S->C: SIP/2.0 200 OK
      Location: sip://bob@host.example.com ;service=IP,voice-mail
                ;media=audio ;duplex=full ;q=0.7
      Location: phone://1-415-555-1212 ; service=ISDN;mobility=fixed;
                language=en,es,iw ;q=0.5
      Location: phone://1-800-555-1212 ; service=pager;mobility=mobile;
                duplex=send-only;media=text; q=0.1
```

Alternatively, Bob could have returned a description of

```
C->S: OPTIONS bob@example.com SIP/2.0
      From: alice@anywhere.org (Alice)
      To: bob@example.com (Bob)
      Accept: application/sdp
```

```
S->C: SIP/2.0 200 OK
      Content-Length: 81
      Content-Type: application/sdp

      v=0
      m=audio 0 RTP/AVP 0 1 3 99
      m=video 0 RTP/AVP 29 30
      a:rtpmap:98 SX7300/8000
```

10 Compact Form

When SIP is carried over UDP with authentication and a complex session description, it may be possible that the size of a request or reply is larger than the MTU. To reduce this problem, a more compact form of SIP is also defined by using alternative names for common header fields. These short forms are NOT abbreviations, they are field names. No other abbreviations are allowed.

short field name	long field name	note
c	Content-Type	
e	Content-Encoding	
f	From	
i	Call-ID	
l	Content-Length	
m	Location	from “moved”
s	Subject	
t	To	
v	Via	

Thus the header in section 9.1 could also be written:

```
INVITE schooler@vlsi.caltech.edu SIP/2.0
v:SIP/2.0/UDP 239.128.16.254 16
v:SIP/2.0/UDP 131.215.131.131
v:SIP/2.0/UDP 128.16.64.19
f:mjh@isi.edu
t:schooler@cs.caltech.edu
i:62729-27@128.16.64.19
c:application/sdp
l:187

v=0
o=user1 53655765 2353687637 IN IP4 128.3.4.5
```

```
s=Mbone Audio
i=Discussion of Mbone Engineering Issues
e=mbone@somewhere.com
c=IN IP4 224.2.0.1/127
t=0 0
m=audio 3456 RTP/AVP 0
```

Mixing short field names and long field names is allowed, but not recommended. Servers MUST accept both short and long field names for requests. Proxies MUST NOT translate a request between short and long forms if authentication fields are present.

11 SIP Transport

SIP is defined so it can use either UDP or TCP as a transport protocol.

11.1 Achieving Reliability For UDP Transport

11.1.1 General Operation

SIP assumes no additional reliability from IP. Requests or replies may be lost. A SIP client SHOULD simply retransmit a SIP request periodically with timer $T1$ (default value of $T1$: once a second) until it receives a response, or until it has reached a set limit on the number of retransmissions. The default limit is 20.

SIP requests and replies are matched up by the client using the Call-ID header field; thus, a server can only have one outstanding request per call at any given time.

HS: A transaction or request ID would remove this limitation.

If the reply is a provisional response, the initiating client SHOULD continue retransmitting the request, albeit less frequently, using timer $T2$. The default retransmission interval $T2$ is 5 seconds.

After the server sends a final response, it cannot be sure the client has received the response, and thus SHOULD cache the results for at least 30 seconds to avoid having to, for example, contact the user or user location server again upon receiving a retransmission.

11.1.2 INVITE

Special considerations apply for the INVITE method.

1. After receiving an invitation, considerable time may elapse before the server can determine the outcome. For example, the called party may be "rung" or extensive searches may be performed, so delays can reach several tens of seconds.

2. It is possible that the invitation request reaches the callee and the callee is willing to take the call, but that the final response (200 OK, in this case) is lost on the way to the caller. If the session still exists but the initiator gives up on including the user, the contacted user has sufficient information to be able to join the session. However, if the session no longer exists because the invitation initiator "hung up" before the reply arrived and the session was to be two-way, the conferencing system should be prepared to deal with this circumstance.

3. If a telephony user interface is modeled or if we need to interface to the PSTN, the caller will provide "ringback", a signal that the callee is being alerted. Once the callee picks up, the caller needs to know so that it can enable the voice path and stop ringback. The callee's response to the invitation could get lost. Unless the response is transmitted reliably, the caller will continue to hear ringback while the callee assumes that the call exists.

4. The client has to be able to terminate an on-going request, e.g., because it is no longer willing to wait for the connection or search to succeed. One cannot rely on the absence of request retransmission, since the server would have to continue honoring the request for several request retransmission periods, that is, possible tens of seconds if only one or two packets can be lost.

The first problem is solved by indicating progress to the caller: the server returns a provisional response indicating it is searching or ringing the user.

The server retransmits the final response at intervals of $T3$ (default value of $T3$ = 2 seconds) until it receives a CONNECTED request for the same Call-ID or until it has retransmitted the final response 10 times. The CONNECTED request is acknowledged only once. If the request is syntactically valid and the Request-URI matches that in the INVITED request with the same Call-ID, the server answers with status code 200, otherwise with status code 400.

Fig. 4 and 5 show the client and server state diagram for invitations.

11.2 Connection Management for TCP

A single TCP connection can serve one or more SIP transactions. A transaction contains zero or more provisional responses followed by exactly one final response.

The client MAY close the connection at any time. Closing the connection before receiving a final response signals that the client wishes to abort the request.

The server SHOULD NOT close the TCP connection until it has sent its final response, at which point it MAY close the TCP connection if it wishes to. However, normally it is the client's responsibility to close the connection.

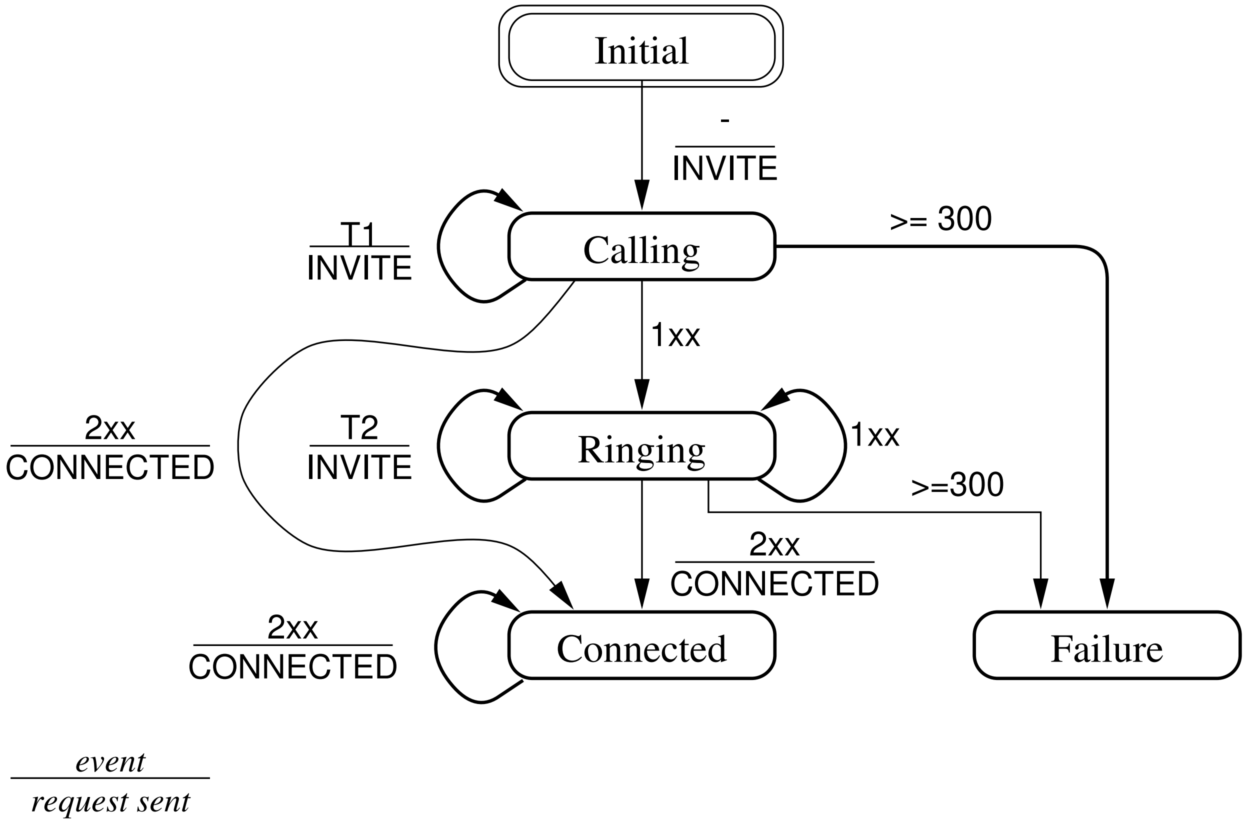

Figure 4: State transition diagram of client for INVITE method

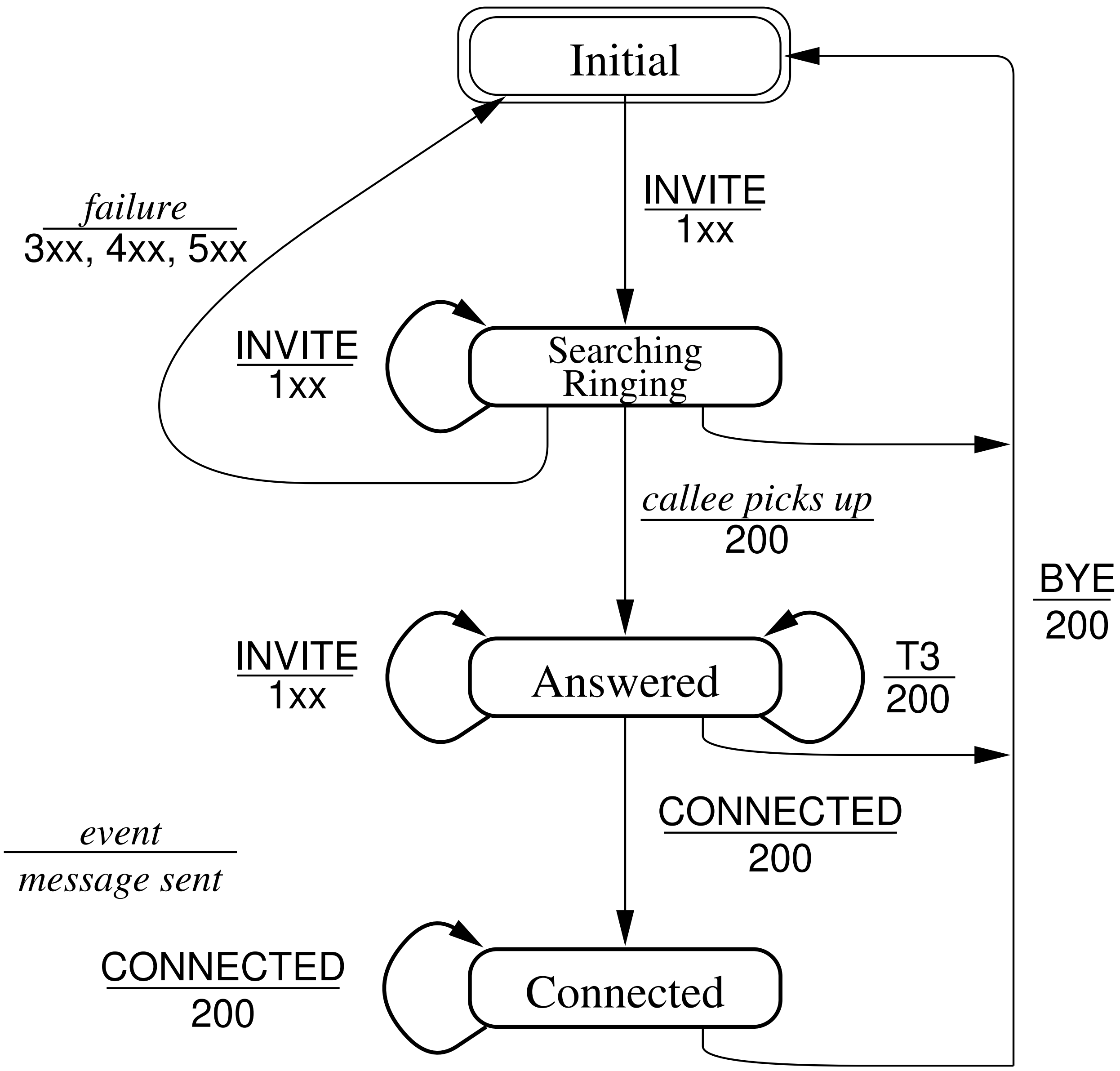

Figure 5: State transition diagram of server for INVITE method

If the server leaves the connection open, and if the client so desires it may re-use the connection for further SIP requests or for requests from the same family of protocols (such as HTTP or stream control commands).

12 Behavior of SIP Servers

This section describes behavior of a SIP server in detail. Servers can operate in proxy or redirect mode. Proxy servers can "fork" connections, i.e., a single incoming request spawns several outgoing (client) requests.

A proxy server always inserts a Via header field containing their own address into requests it issues that are caused by an incoming request.

We define an "A–B proxy" as a proxy that receives SIP requests over transport protocol A and issues requests, acting as a SIP client, using transport protocol B. If not stated explicitly, rules apply to any combination of transport protocols. For conciseness, we only describe behavior with UDP and TCP, but the same rules apply for any unreliable datagram or reliable protocol, respectively.

The detailed connection behavior for UDP and TCP is described in Section 11.

12.1 Redirect Server

A redirect server does not issue any SIP requests of its own. It can return a response that accepts, refuses or redirects the request. After receiving a request, a redirect server proceeds through the following steps:

1. If the request cannot be answered immediately (e.g., because a location server needs to be contacted), it returns one or more provisional responses.
2. Once the server has gathered the list of alternative locations or has decided to accept or refuse the call, it returns the final response. This ends the SIP transaction.

The redirect server maintains transaction state for the whole SIP transaction. Servers in user agents are redirect servers.

12.2 Proxies Issuing Single Unicast Requests

Proxies in this category issue at most a single unicast request for each incoming SIP request, that is, they do not "fork" requests. Servers may choose to always operate in the mode described in Section 12.3.

12.2.1 UDP–UDP Proxy Server

The UDP–UDP server can forward the request and any responses. It does not have to maintain any state for the SIP transaction. UDP reliability is assured by the next redirect server in the server chain.

12.2.2 UDP–TCP Proxy Server

A proxy server issuing a single request over TCP maintains state for the whole SIP transaction indexed by the Call-ID.

If it receives a UDP retransmission of the same request for an existing session, it retransmits the last response received from the TCP side. Any changes in the message body compared to the last request for the Call-ID are silently ignored. (Otherwise, the proxy would have to remember and compare the message body; this also violates the notion of a SIP transaction. !!!TBD!!!) The server SHOULD cache the final response for a particular Call-ID after the SIP transaction on the TCP side has completed.

After the cache entry has been expired, the server cannot tell whether an incoming request is actually a retransmission of an older request, where the TCP side has terminated. It will treat it as a new request.

12.3 Proxy Server Issuing Several Requests

All requests carry the same Call-ID. For unicast, each of the requests has a different (host-dependent) Request-URI. For multicast, a single request is issued, likely with a host-independent Request-URI. A client receiving a multicast query does not have to check whether the *host* part of the Request-URI matches its own host or domain name. To avoid response implosion, servers SHOULD NOT answer multicast requests with a 404 (Not Found) status code. Servers MAY decide not to answer multicast requests if their response would be 5xx.

The server MAY respond to the request immediately with a "100 Trying" response; otherwise it MAY wait until either the first response to its requests or the UDP retransmission interval.

The following pseudo-code describes the behavior of a proxy server issuing several requests in response to an incoming request. The function `request(`a`)` sends a SIP request to address a. `await_response()` waits until a response is received and returns the response. `request_close(`a`)` closes the TCP connection to client with address a; this is optional. `response(`s`, `l`, `L`)` sends a response to the client with status s and list of locations L, with l entries. `ismulticast()` returns 1 if the location is a multicast address and zero otherwise. The variable `timeleft` indicates the amount of time left until the maximum response time has expired. The variable `recurse` indicates whether the server will recursively try addresses returned through a 3xx response. A server MAY decide to recursively try only certain addresses, e.g., those which are within the same domain as the proxy server. Thus, an initial multicast request may trigger additional unicast requests.

```
int N = 0;            /* number of connection attempts */
```

```
address_t address[];  /* list of addresses */
location[];           /* list of locations */
int heard = 0;        /* number of sites heard from */
int class;            /* class of status code */
int best = 1000;      /* best response so far */
int timeleft = 120;   /* sample timeout value */
int loc = 0;          /* number of locations */
struct {              /* response */
  int status;         /* response status */
  char *location;     /* redirect locations */
  address_t a;        /* address of respondent */
} r;
int i;

if (multicast) {
  request(address[0]);
} else {
  N = /* number of addresses to try */
  for (i = 0; i < N; i++) {
    request(address[i]);
  }
}

while (timeleft > 0 && (heard < N || multicast)) {
  r = await_response();
  class = r.status / 100;

  if (class >= 2) {
    heard++;
    if (tcp) request_close(a);
  }

  if (class == 2) {
    best = r.status;
    break;
  }
  else if (class == 3) {
/* A server may optionally recurse.  The server MUST check whether
 * it has tried this location before and whether the location is
 * part of the Via path of the incoming request.  This check is
 * omitted here for brevity. Multicast locations MUST NOT be
     * returned to the client if the server is not recursing.
     */
    if (recurse) {
      multicast = 0;
```

```
          N++;
          request(r.location);
        } else if (!ismulticast(r.location)) {
          locations[loc++] = r.location;
          best = r.status;
        }
      }
      else if (class == 4) {
        if (best >= 400) best = r.status;
      }
      else if (class == 5) {
        if (best >= 500) best = r.status;
      }
      else if (class == 6) {
        best = r.status;
        break;
      }
    }
    /* We haven't heard anything useful from anybody. */
    if (best == 1000) {
      best = 404;
    }
    if (best/100 != 3) locs = 0;
    response(best, locs, locations);
```

When operating in this mode, a proxy server MUST ignore any responses received for Call-IDs that it does not have a pending transaction for. (If server were to forward responses not belonging to a current transaction using the Via field, the requesting client would get confused if it has just issued another request using the same Call-ID.)

13 Security Considerations

13.1 Confidentiality

Unless SIP transactions are protected by lower-layer security mechanisms such as SSL [**?**], an attacker may be able to eavesdrop on call establishment and invitations and, through the Subject header field or the session description, gain insights into the topic of conversation.

13.2 Integrity

Unless SIP transactions are protected by lower-layer security mechanisms such as SSL [**?**], an active attacker may be able to modify SIP requests.

13.3 Access Control

SIP requests are not authenticated unless the SIP Authorization and WWW-Authenticate headers are being used. The strengths and weaknesses of these authentication mechanisms are the same as for HTTP.

13.4 Privacy

User location and SIP-initiated calls may violate a callee's privacy. An implementation SHOULD be able to restrict, on a per-user basis, what kind of location and availability information is given out to certain classes of callers.

A Summary of Augmented BNF

In this specification we use the Augmented Backus-Naur Form notation described in [19]. For quick reference, the following is a brief summary of the main features of this ABNF.

"abc"
: The case-insensitive string of characters "abc" (or "Abc", "aBC", etc.);

%d32
: The character with ASCII code decimal 32 (space);

*term
: zero of more instances of term;

3*term
: three or more instances of term;

2*4term
: two, three or four instances of term;

[term]
: term is optional;

{ term1 term2 term3 }
: set notation: term1, term2 and term3 must all appear but their order is unimportant;

term1 | term2
either term1 or term2 may appear but not both;

#term
a comma separated list of term;

2#term
a comma separated list of term containing at least 2 items;

2#4term
a comma separated list of term containing 2 to 4 items.

Common Tokens

Certain tokens are used frequently in the BNF this document, and not defined elsewhere. Their meaning is well understood but we include it here for completeness.

CR	=	%d13	; carriage return character
LF	=	%d10	; line feed character
CRLF	=	CR LF	; typically the end of a line
SP	=	%d32	; space character
TAB	=	%d09	; tab character
LWS	=	*(SP | TAB)	; linear whitespace
DIGIT	=	"0" .. "9"	; a single decimal digit

Changes

Since version -01, the following things have changed:

- Added personal note to "Searching" section indicating that 6xx codes may not be necessary. Added figures.
- Initial author's note removed; dated.
- Introduction rewritten to give quick, concise overview as to what SIP does.
- Conference control (tight vs. loose) seems less and less appropriate. All share some state such as notions of membership; some (ITU versions) tend to keep it in a central server, others distribute it. Some state is synchronized at larger timescales than other. (After all, even a server won't know if a participant disconnects from the network until TCP keep-alive, if any, kicks in.)
- Added list of related protocols to emphasize that this is part of a whole architecture.

- Terminology: user always reminds me of controlled substances; thus, this term is avoided where better terminology exists. Since this protocol sits at the boundary between traditional Internet and telephony services, some of the terminology familiar in that realm is introduced.
- Terminology: user location server replaced by redirect server, since a proxy server may also invoke user location. Also, the actual user location server (e.g., an LDAP, ULS or similar directory) may be invoked using protocols other than SIP.
- Rearranged ordering of address resolution to correspond to host requirements for MX and suggestions in DNS SRV RFC. Adding note about caching and socket implementation. Added note about using SMTP EXPN to get an alternate address.
- Defined SIP transaction, provisional and final responses.
- Assigned values to timeout parameters; otherwise, there will be unnecessary retransmissions between different implementations.
- Retransmission was greatly simplified; there does not seem to be a need for all the rules governing transitions between TCP and UDP domains. A proxy should look just like a server to one side and like a client to the other. Proxies need to maintain transaction state in any event since they need to remember where they forwarded the last SIP request to (Confirm wouldn't work otherwise, for example.). Invoking a location service may yield inconsistent results, introduces additional failure modes (what if the location server is temporarily unavailable?), increases delay and processing overhead. UDP–UDP proxies can still be built without state; they just forward packets and responses. Proxies with TCP on one and UDP on the other side will have to act like a normal UDP server and issue 100 responses.
- Removed redundancies and contradictions from request and response definitions (space vs. SP, duplicate CRLF definition, recursive request-header, . . .).
- Added the experimental methods CONNECTED, REGISTER, UNREGISTER and BYE.
- Re-engineered the invitation reliability mechanism to use a separate confirmation message.
- Tentative increase of MTU to 1500 bytes, as per discussion in Stevens.
- Added Reach, Organization, Subject, Priority, Authorization, WWW-Authentication headers for improved call handling. WWW "basic" authentication isn't great, but it is widely deployed and probably sufficient for giving out "private" telephone numbers, particularly those where the callee incurs a charge. (I want to be able to give somebody a password to call my 800 number via an Internet gateway; authenticating who that person is requires that I modify a script on my server to add another distinguished name to the list of allowable callees.)
- Renamed Reason to Warning (to align with HTTP) header since the response line already offers a failure reason. Unfortunately, listing several failures is not all that helpful since the calling party cannot determine which of the media within the description causes the difficulty or whether it was the set of media as a whole, but it may give the user agent some indication as to what's going on.

- SEP and CRLF in headers removed, since this is always implied between items. Missing ":" added. CRLF was already in the message definition. Also, unlike RFC 822 and HTTP, the definition did not allow spaces between the field name and the colon.
- Added (reluctantly) password to URL. It's no worse than ftp and needed to easily call from a secure web page, without having to type in a password manually.
- Added port to SIP URL to specify non-standard port.
- CAPABILITIES to OPTIONS for closer alignment with HTTP and RTSP;
- Path to Via for closer alignment with HTTP and RTSP;
- Content type meta changed to application, since "meta" doesn't exist as a top-level Internet media type.
- Formatting closer to HTTP and RTSP.
- Explain relationship to H.323.

B Open Issues

RELIABLE: How to provide reliability?

BYE: Use of BYE method?

REGISTER: Use of REGISTER method?

H.323: Interaction with H.323 and H.245.

TRANSACTION: Should we have a transaction id in addition to a call ID? Call-IDs are for the end system, but a transaction ID is for a single SIP exchange. This is useful for Internet telephony, where a single call may trigger several transactions.

C Acknowledgments

We wish to thank the members of the IETF MMUSIC WG for their comments and suggestions. This work is based, inter alia, on [23, 24]. Parameters of the terminal negotiation mechanism were influenced by Scott Petrack's CMA design.

D Authors' Addresses

Mark Handley

USC Information Sciences Institute
c/o MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
USA
electronic mail: mjh@isi.edu

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027
USA
electronic mail: schulzrinne@cs.columbia.edu

Eve Schooler
Computer Science Department 256-80
California Institute of Technology
Pasadena, CA 91125
USA
electronic mail: schooler@cs.caltech.edu

References

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

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP) – version 1 functional specification," Internet Draft, Internet Engineering Task Force, June 1997. Work in progress.

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

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

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

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

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

[8] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

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

[10] C. Partridge, "Mail routing and the domain system," STD 14, RFC 974, Internet Engineering Task Force, Jan. 1986.

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

[12] P. Mockapetris, "Domain names - implementation and specification," STD 13, RFC 1035, Internet Engineering Task Force, Nov. 1987.

[13] R. Braden, "Requirements for internet hosts - application and support," STD 3, RFC 1123, Internet Engineering Task Force, Oct. 1989.

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

[15] W. Yeong, T. Howes, and S. Kille, "Lightweight directory access protocol," RFC 1777, Internet Engineering Task Force, Mar. 1995.

[16] T. Berners-Lee, "Universal resource identifiers in WWW: a unifying syntax for the expression of names and addresses of objects on the network as used in the world-wide web," RFC 1630, Internet Engineering Task Force, June 1994.

[17] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource locators (URL): Generic syntax and semantics," Internet Draft, Internet Engineering Task Force, May 1997. Work in progress.

[18] T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform resource locators (URL)," RFC 1738, Internet Engineering Task Force, Dec. 1994.

[19] D. Crocker, "Augmented BNF for syntax specifications: ABNF," Internet Draft, Internet Engineering Task Force, Oct. 1996. Work in progress.

[20] J. Mogul and S. Deering, "Path MTU discovery," RFC 1191, Internet Engineering Task Force, Nov. 1990.

[21] W. R. Stevens, *TCP/IP illustrated: the protocols*, vol. 1. Reading, Massachusetts: Addison-Wesley, 1994.

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

[23] E. M. Schooler, "Case study: multimedia conference control in a packet-switched teleconferencing system," *Journal of Internetworking: Research and Experience*, vol. 4, pp. 99–120, June 1993. ISI reprint series ISI/RS-93-359.

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