Internet Engineering Task Force MMUSIC WG
INTERNET-DRAFT
M. Handley, H. Schulzrinne, E. Schooler
ISI,Columbia,Caltech
draft-ietf-mmusic-sip-01.ps
2nd Dec 1996
Expires: 2nd June 1997

SIP: Session Initiation Protocol

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, providing support for either loosely or tightly controlled sessions. 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 Multiparty 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.

1. Authors' Note

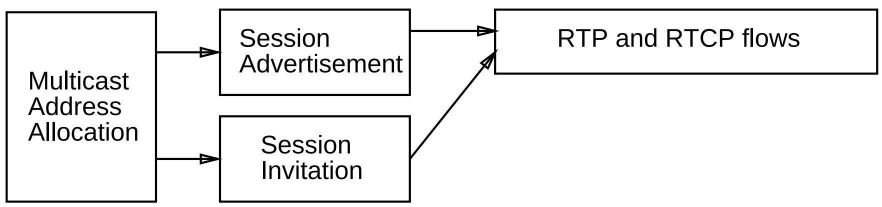
This document is the result of a merger of the Session Invitation Protocol (draft-ietf-mmusic-sip-00.txt) and the Simple Conference Invitation Protocol (draft-ietf-mmusic-scip-00.txt), and of an attempt to make SIP more generic and to fit into a more flexible infrastructure which includes companion protocols including SDP, HTTP and RTSP.

2. Introduction

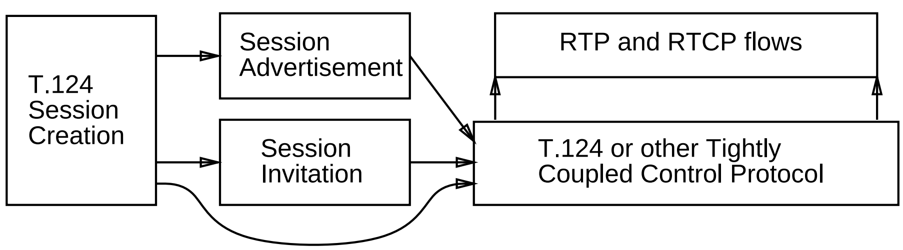
There are two basic ways to locate and to participate in a multimedia session:

- The session is advertised, users see the advertisement, then join the session address to participate.
- Users are invited to participate in a session, which may or may not already be advertised.

The Session Description Protocol (SDP) together with the Session Announcement Protocol (SAP), provide a mechanism for the former [1] [2]. This document presents the Session Initiation Protocol (SIP), to perform the latter. SIP also can use SDP to specify what is meant by a session.



Lightweight Session Lifecycle



Tightly Coupled Session Lifecycle

Figure 1

We make the design decision that how a user discovers that a session exists is orthogonal to a session's conference control model. Figure 1 shows a potential place for SIP in the lifecycle of both lightweight sessions and in more tightly-coupled conferencing. Note that the Session Initiation Protocol and the Session Announcement Protocol may be invoked or re-invoked at later stages in a session's lifecycle.

The Session Initiation Protocol is also intended to be used to invite servers into sessions. Examples might be where a recording server can be invited to participate in a live multimedia session to record that session, or a video-on-demand server can be invited to play a video stream into a live multimedia conference. In such cases we would like SIP to lead gracefully into the control protocol to control recording and playback on such servers.

We also make the design decision that inviting a user to participate in a session is independent of quality of service (QoS) guarantees for that session. Such QoS guarantees (if they are required) may be dependent on the full membership of the session, and this may or may not be known to the agent performing session invitation.

2.1. Terminology

This document uses the same words as RFC 1123 for defining the significance of each particular requirement. These words are:

must:
This word or the adjective ‘‘required’’ means that the item is an absolute requirement of the specification.

should:
This word or the adjective ‘‘recommended’’ means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

may:
This word or the adjective ‘‘optional’’ means that this item is truly optional. One implementation may choose to include the item because a particular application requires it or because it enhances the product, for example, another implementation may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the *must* requirements for the protocols it implements. An implementation that satisfies all the *must* and all the *should* requirements for the protocols it implements is said to be ‘‘unconditionally compliant’’; one that satisfies all the *must* requirements but not all of the *should* requirements for the protocols it implements is said to be ‘‘conditionally compliant’’.

2.2. Glossary

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 HTTP.

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

Initiator:
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 attempt to contact a user (or service) to request that they participate in a session.

Invitee, Invited User:
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 which returns one or more possible locations for the user or service without contacting that user or service directly.

Location service:
A service used by a location server to obtain information about a user's possible location.

Proxy, Proxy server:
An intermediary program which 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, with possible translation, on to other servers. A proxy must interpret, and, if necessary, rewrite a request message before forwarding it.

Server:
An application program that accepts connections in order to service requests by sending back responses. A server may be the called user agent, a proxy server, or a location server.

User Agent, Called User Agent:
The server application which contacts the invitee to inform them of the invitation, and to return a reply.

Any given program may be capable of acting both as a client and a server. A typical multimedia conference controller would act as a client to initiate calls or invite others to conferences and as a server to accept invitations.

3. General Requirements

SIP is a Session Initiation Protocol. It is not a conference control protocol. SIP can be used to perform a search for a user or service and to request that that user or service participate in a session.

Once SIP has been used to initiate a multimedia session SIP's task is finished. **There is no concept of a SIP session** (as opposed to a SIP search for a user or service). If whatever conference control mechanism is used in the session needs to add or remove a media stream, SIP may be used to perform this task, but again, once the information has been successfully conveyed to the participants, SIP is the no longer involved.

SIP must be able to utilise both UDP and TCP as transport protocols.

From a performance point of view, UDP is preferable as it allows the application to more carefully control the timing of messages, it allows parallel searches without requiring connection state for each outstanding request, and allows the use of multicast.

From a pragmatic point of view, TCP allows easier passage through existing firewalls, and with appropriate protocol design, allows common SIP, HTTP and RTSP servers.

When TCP is used, SIP can use either one or more than one connection 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.

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 only used for session initiation, it is believed that the additional overhead of using a text-based protocol is not significant.

Unlike control protocols, there is minimal shared-state in SIP - in a minimal implementation the initiator maintains all the state about the current attempt to locate and contact a user or service - servers or proxies can be stateless (although they don't have to be). All the state needed to get a response back from a server to the initiator is carried in the SIP request itself - this is also necessary for loop prevention.

Authors Note: Whilst redesigning SIP, we have attempted to ensure that it can has a clear interaction with the currently evolving Real-Time Stream Control Protocol. Whether RTSP will adopt an approach that makes this meaningful remains to be seen.

4. Addressing

SIP is a protocol that exchanges messages between peer *user agents* or *proxies* for user agents. We assume the user agent is an application that acts on behalf of the user it represents (thus it is sometimes described as a *client* of the user) and that is co-resident with that user. A proxy for a user agent serves as a forwarding mechanism or bridge to the actual location of the user agent. We also refer to such proxies as *conference address servers*.

In the computer realm, the equivalent of a personal telephone number combines the user's login id (`mjh`) with a machine host name (`metro.isi.edu`) or address (`128.16.64.78`). A user's location-specific address can be obtained out-of-band, can be learned via existing media agents, such as vat (e.g., Mbone Audio channel), can be included in some mailers' message headers, or can be recorded during previous invitation interactions.

However, users also publish several well-known addresses that are relatively location-independent, such as email or web home-page addresses. Rather than require that users provide their specific network locales, we can take advantage of email and web addresses as being (relatively) memorable, and also leverage off the Domain Name Service (DNS) to provide a first stage location mechanism. Note that an email address (`M.Handley@cs.ucl.ac.uk`) is usually different from the combination of a specific machine name and login name (`mjh@mercury.lcs.mit.edu`). SIP should allow both forms of addressing to be used, with the former requiring a conference address server to locate the user.

One perceived problem of email addressing is that it is possible to guess peoples' addresses and thus the system of unlisted (in the telephone directory) numbers is more of a problem. However, this really only provides security through obscurity, and real security is better provided through authentication and call screening.

5. Call Setup

Call setup is a multi-phase procedure. In the first phase, the requesting client tries to ascertain the address where it should contact the remote user agent or user agent proxy. The local client checks if the user address is location-specific. If so, then that is the address used for the remote user agent. If not, the requesting client looks up the domain part of the user address in the DNS. This provides one or more records giving IP addresses. If a new service (SRV) resource record [4] is returned giving a conference address server, then that is the address to contact next. If no relevant resource record is returned, but an A record is returned, then that is the address to contact next. If neither a resource record or an A record is returned, but an MX record is returned, then the mail host is the address to contact next.

Presuming an address for the invitee is found from the DNS, the second and subsequent phases basically implement a request-response protocol. A session description (typically using SDP format) is sent to the contact address with an invitation for the user to join the session.

This request may be sent over a TCP connection or as a single UDP datagram (the format of both is the same and is described later), and is sent to a well-known port.

If a user agent or conference server is listening on the relevant port, it can send one of the responses below. If no server or agent is listening, an ICMP port-unreachable response will be triggered which should cause the TCP connection setup to fail or cause a UDP send failure on retransmissions.

5.1. Locating a User

It is expected that a user is situated at one of several frequented locations. These locations can be dynamically registered with a conference address server for a site (for a local area network or organization), and incoming connections can be routed simultaneously to all of these locations if so desired. It is entirely up to the conference address server whether the server issues proxy requests for the requesting user, or if the server instructs the client to redirect the request.

In general a reply MUST be sent by the same mechanism that the request was sent by. Hence, if a request was unicast, then the reply MUST be unicast back to the requester; if the request was multicast, the reply MUST be multicast to the same group to which the request was sent; if the request was sent by TCP, the reply MUST be sent by TCP.

In all cases where a request is forwarded onwards, each host relaying the message SHOULD add its own address to the path of the message so that the replies can take the same path back, thus ensuring correct operation through compliant firewalls and loop-free requests. On the reply path, these routing headers MUST be removed as the reply retraces the path, so that routing internal to sites is hidden. When a multicast request is made, first the host making the request, then the multicast address itself are added to the path.

6. Message Formats

All messages are text-based. When sent over TCP or UDP, multiple requests can be carried in a single TCP connection or UDP datagram. UDP Datagrams should not normally exceed the path MTU in size if it is known, or 1 KByte if the MTU is unknown.

6.1. Session Invitations

An example of a session invitation is shown below. It is divided into a request and a number of header fields. The request is of the form:

<method> *<request-id>* `SIP/2.0`

The method may be either `INVITE` or `CAPABILITY`. The request 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.

6.1.1. Methods

The following methods are defined:

INVITE
The user or service is being invited to participate in the session. The session description given must be completely acceptable for a "**200 OK**" response to be given. This method MUST be supported by a SIP server.

CAPABILITY
The user or service 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.

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 user agent should cause a "501 Not Implemented" response to be returned.

6.1.2. Header Fields

SIP header fields are similar to MIME header fields in both syntax and semantics. In general the ordering of the header fields is not of importance (with the exception of Path fields, see below) but proxies MUST NOT reorder or otherwise modify header fields other than by adding a new Path field. This allows an authentication field to be added after the Path fields that will not be invalidated by proxies. Field names are not case-sensitive, although their values may be.

Content-Length, Content-Type, To, From, Path header fields are compulsory. Other fields may be added as required. Header fields MUST be separated by a single linefeed character. The header MUST be separated from the payload by an emply line (two linefeed characters).

A compact form of these codes is also defined in section 6.3 for use over UDP when the request has to fit into a single packet and size is an issue.

- The "Path:" 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. Initiators MUST add their own Path field to each request. This Path field MUST be the first field in the request. Subsequent proxies SHOULD each add their own additional Path field which MUST be added before any existing Path fields. When a reply passes through a proxy on the reverse path, that proxies Path field MUST be removed from the reply.

  The format for a Path field is:

  `Path:`<*net-type*> <*address-type*> <*protocol*> <*address*> <*sequence*> `[`<*ttl*>`]`

  Net type is typically `IN` (Internet) and address-type is typically `IP4` or `IP6` for IP version 4 and IP version 6 respectively. Protocol may be `UDP` or `TCP`. Sequence is a request sequence number which typically starts at 1 and is incremented each time the request is resent by this host. TTL is included if the address is a multicast address.

- Authentication fields provide a digital signature of the remaining fields for authentication purposes. They are *not yet defined*, but when they are defined they MUST be added to the header after the Path fields and before the rest of the fields.

- The request header MUST contain both a "From:" field , indicating the invitation initiator, and a "To:" field , specifying the invited user. Both of these fields MUST be machine-usable, as defined my `mailbox` in RFC 822 (as updated by RFC 1123). Only a single initiator and a single invited user are allowed to be specified in a single SIP request.

- The session description payload gives details of the session the user is being invited to join. It's format MUST be given by the "Content-type:" header field, and the payload length in bytes MUST be given by the "Content-length" header field. If the payload has undergone any encoding (such as compression) then this MUST be indicated by the "Content-encoding:" header field, otherwise " Content-encoding:" MUST be omitted.

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

```
INVITE 128.16.64.19/65729 SIP/2.0
Path:IN IP4 UDP 239.128.16.254 1 16
Path:IN IP4 UDP 131.215.131.131 1
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Content-type:meta/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 path fields (`Path:`...) 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` (specifically it was initiated from 128.16.64.19, as can be seen from the path field) and the user being invited is `schooler@cs.caltech.edu`.

In this case, the session description (as stated in the "`Content-type:`" field) is a Session Description Protocol (SDP) description as defined in the companion draft [1].

The header is terminated by an empty line (two linefeed characters) and is followed by the session description payload.

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.

6.2. Responses

6.2.1. Response Codes

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 are marked with an asterisk, and have codes x50 upwards to avoid clashes with future HTTP response codes, or 6xx which are not used by HTTP. The default behaviour for unknown response codes is given for each category of codes.

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.

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

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

Successful 2xx

The request was successful and MUST terminate a search.

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

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.

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 reponse 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 request's Path as the user's new location.

302 Moved Temporarily
The requesting client should retry on the new address(es) given by the `Location:` field. A 302 response MUST NOT suggest any of the hosts in the request's Path as the user's new location.

350* Alternative Service
The call was not successful, but alternative services are possible. The alternative services are described in the body of the reply.

Request Failure 4xx

4xx responses are definite failure responses that MUST terminate the existing search for a user or service. They SHOULD NOT be retried immediately without modification.

400 Bad Request
The request could not be understood due to malformed syntax.

401 Unauthorized
The request requires user authentication.

402 Payment Required
Reserved for future use.

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

404 Not Found
The server has definitive information that the user does not exist at the domain specified.

406 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.

450* Decline
The user's machine was successfully contacted but the user explicitly does not wish to participate.

451* Busy
The user's machine was successfully contacted but the user is busy, or the user does not wish to participate (the ambiguity is intentional).

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.

500 Server Internal Error
The server encountered an unexpected condition that prevented it from fulfilling the request.

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 recognise the request method and is not capable of supporting it for any user.

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.

Search Responses 6xx

6xx responses are failure responses given whilst trying to locate the specified user or service. They are not definitive failures, and SHOULD NOT terminate the search if other possible locations remain untried.

600* Search Failure
The user agent or proxy server understood the user's address, but the request was unsuccessful in contacting the user. A proxy might return this error towards the initiator if an attempt to contact a server failed for an unknown reason.

601* Not known here
The call was unsuccessful because the user or service was not known at the address called. This is not a definitive failure - the address may be valid at another server.

602* Not currently here
The call was unsuccessful because although the the user or service was known at the address called, the user or service is not currently located at this address. This is not a definitive failure - the user may be contactable at another server.

603* Alternative Address
The call was unsuccessful because the user or service is not available at this location, but one or more alternative non-definitive locations are suggested to try in addition to any that may already be being tried. A 603 response MUST NOT suggest any of the hosts in the request's Path as an alternative location.

6.2.2. Normal Replies

An example reply 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 path fields are taken from the request, and entries are removed hop by hop as the reply retraces the request's path. A new authentication field is added by the invited user's agent if required. The session ID is taken directly from the original request, along with the request header. The original sense of "`From:`"

field is preserved (i.e it's the session originator).

In addition, a "`Contact-host:`" field is added giving details of the host the user was located on, or alternatively the relevant proxy contact point which should be reachable from the invitation initiator's host.

```
SIP/2.0 200 128.16.64.19/65729
Path:IN IP4 UDP 239.128.16.254 1 16
Path:IN IP4 UDP 131.215.131.131 1
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Contact-host:IN IP4 131.215.131.147
Content-length:0
```

This same format is used for replies for other categories of reply, except that some of then may require payloads to be carried.

If the invited user's agent requires confirmation of receipt of a "200 OK" reply, it may optionally add an additional "Confirm:required' field to the body of the message specifying that an acknowledgementy is required. This is only permitted with category 2xx replies. An example is:

```
SIP/2.0 200 128.16.64.19/65729
Path:IN IP4 UDP 239.128.16.254 1 16
Path:IN IP4 UDP 131.215.131.131 1
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Contact-host:IN IP4 131.215.131.147
Confirm:required
Content-length:0
```

In response to such a request, the invitation initiators agent should retransmit its request with an additional "Confirm:" field, with the value "true" or "false" stating whether the session still exists or no longer exists respectively (see section 7.1 for details). An example of an confirmation request is:

```
INVITE 128.16.64.19/65729 SIP/2.0
Path:IN IP4 UDP 239.128.16.254 1 16
Path:IN IP4 UDP 131.215.131.131 1
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Confirm:true
Content-type:meta/sdp
Content-Length:187

v=0
o=user1 2353655765 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
```

Such confirmations are still useful when TCP transport is used as they provide application level confirmation rather than transport level confirmation. If they are not used, it is possible that a "200 OK" response may be received after the application making the call has timed out the call and exited.

6.2.3. Redirects

"603 alternative address" replies and 301 and 302 moved replies should specify another location using the Location: field.

An example of a "603 alternative address" reply is:

```
SIP/2.0 603 128.16.64.19/65729
Path:IN IP4 UDP 131.215.131.131 1
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Location:IP IP4 239.128.16.254 16
Content-length:0
```

In this example, the proxy (131.215.131.131) is being advised to contact the multicast group 239.128.16.254 with a ttl of 16. 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.

For unicast 603 redirects, a proxy MAY query the suggested location itself or send MAY the redirect on back towards the client. For multicast 603 redirects, a proxy SHOULD query the multicast address itself rather than sending the redirect back towards the client as multicast may be scoped and this allows a proxy within the appropriate scope regions to make the query.

For 301 or 302 redirects, a proxy SHOULD send the redirect on back towards the client and terminate any other searchs it is performing for the same request. Multicast 301 or 302 redirects MUST NOT be generated.

Alternative Services

An example of an "350 Alternative Service" reply is:

```
SIP/2.0 350 128.16.64.19/32492/2
Path:IN IP4 UDP 131.215.131.131 1
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Contact-host:IN IP4 131.215.131.131
Content-type:meta/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 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 contact host (131.215.131.131) with the session description provided. This request should contain a different session id from the one in the original request. An example would be:

```
INVITE 128.16.64.19/32492/3 SIP/2.0
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Content-type:meta/sdp
Content-length: 146

v=0
o=mm-server 2523535 0 IN IP4 128.16.5.31
s=Answering Machine
i=Leave an audio message
c=IN IP4 128.16.64.19
t=0 0
m=audio 12345 RTP PCMU
```

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

6.2.4. Negotiation

A "406 Not Acceptable" reply means that the user wishes to communicate, but cannot support the session described adequately. The "406 Not Acceptable" reply contains a list of reasons why the session described cannot be supported. These reasons can be one or more of:

406.1 Insufficient Bandwidth - the bandwidth specified in the session description or defined by the media exceeds that known to be available.

406.2 Incompatible Protocol - one or more protocols described in the request is not available.

406.3 Incompatible Format - one or more media formats described in the request is not available.

406.4 Multicast not available - the site where the user is located does not support multicast.

406.5 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. If is up to the invitation initiator to decide whether or not to act on a "406 Not Acceptable" reply.

A complex example of a "406 Not Acceptable" reply is:

```
SIP/2.0 406 128.16.64.19/32492/5
Path:IN IP4 UDP 131.215.131.131 1
Path:IN IP4 UDP 128.16.64.19 1
From:mjh@isi.edu
To:schooler@cs.caltech.edu
Contact-host:IN IP4 131.215.131.131
Reason:406.1
Reason:406.3
Reason:406.4
Content-type: meta/sdp
Content-length:

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 256Kbps total bandwidth, and the reply states that only 128Kbps 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.

Invitation initiators MAY choose to treat "406 Not Acceptable" replies as a failure if they wish to do so.

6.3. 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 (or default 1Kbyte limit if the MTU is not known). 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 abbrieviations - they are field names. No other abbrieviations are allowed.

p: same as **Path:**

f: same as **From:**

t: same as **To:**

c: same as **Content-type:**

l: same as **Content-length:**

e: same as **Content-encoding:**

a: same as **Confirm:** (derived from **a**cknowledge)

h: same as **Contact-host:**

m: same as **Location:** (derived from **m**oved)

r: same as **Reason:**

Thus the header in section ?? could also be written:

```
INVITE 128.16.64.19/65729 SIP/2.0
p:IN IP4 UDP 239.128.16.254 1 16
p:IN IP4 UDP 131.215.131.131 1
p:IN IP4 UDP 128.16.64.19 1
f:mjh@isi.edu
t:schooler@cs.caltech.edu
c:meta/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 fieldnames and long fieldnames is allowed, but not recommended. Servers MUST accept both short and long fieldnames for requests. Proxies MUST NOT translate a request between short and long forms if authentication fields are present.

7. SIP Transport

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

UDP has advantages over TCP from a performance point of view, as the SIP application can keep control of the precise timing of retransmissions, and can also make simultaneous call attempts to many potential locations of many users without needing to keep TCP connection state for each connection.

TCP has the advantage that clients are simpler to implement because no retransmission timing code needs to be written and also that it is possible to have a single server serving SIP and HTTP with very little extra code.

With UDP, all the additional reliability code is in the client. It is recommended that servers SHOULD implement both TCP and UDP functionality as the additional server code required is very small.

Clients MAY implement either TCP or UDP transport or both as they see fit.

7.1. Reliability using UDP transport

The Session Invitation Protocol is straightforward in operation. Only the initiating client needs to keep any state regarding the current connection attempt. SIP assumes no additional reliability from IP. Requests or replies may be lost. A SIP client SHOULD simply retransmit a SIP request until it receives a reply, or until it has reached some maximum number of timeouts and retransmissions. If the reply is merely a 1xx Informational progress report, the initiating client SHOULD still continue retransmitting the request, albeit less frequently.

When the remote user agent or server sends a final 2xx or 4xx response (not a 1xx report), it cannot be sure the client has received the response, and thus SHOULD cache the results until a connection setup timeout has occurred to avoid having to contact the user again. The server MAY also choose to cache 3xx or 6xx responses if the cost of obtaining the response outweighs the cost of caching it.

It is possible that a user can be invited successfully, but that the reply that the user was succeffully contacted may not reach the invitation initiator. 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.

One solution is for the initiator to acknowledge the invitee's "200 OK" reply. Although not required, in the case of a successful invitation the invited user's agent can make a confirmation request in its "200 OK" reply. In this case the initiator's agent sends a single request with a reply "`Confirm:true`" if the request was still valid or a reply "`Confirm:false`" if it was not so that a premature hang-up can be detected without a long timeout. Such a confirmation request may be retransmitted by the invited user's agent if it so desired. Confirmation requests can only be made with "200 OK" replies, and only the invitation initiator's agent may issue the actual confirmation.

Only a "`200 OK`" reply warrants such a confirmation handshake, because it is the only situation where user-relevant state may be instantiated anywhere other than at the initiator's client. In all other cases, it is not necessary that state is maintained. In particular, when a server makes multiple proxy requests, "5xx Server Error" and "6xx Search Response" replies do not immediately get passed back to the invitation initiator, and so no end-to-end acknowledgment of a failed request is possible.

7.2. Reliability using TCP transport

TCP is a reliable transport protocol, and so we do not need to define additional reliability mechanisms. However we must define rules for connection closedown under normal operation.

The normal mode of operation is for the client (or proxy acting as a client) to make a TCP connection to the well-known port of a host housing a SIP server. The client then sends the SIP request to the server over this connection and waits for one or more replies. The client MAY close the connection at any time.

The server MAY send one or more 1xx Informational responses before sending a single 2xx, 3xx, 4xx, 5xx or 6xx reply. The server MUST NOT send more than one reply, with the exception of 1xx responses. 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.

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).

The same application-level confirmation rules apply for TCP as for UDP.

8. Searching

A basic assumption of SIP is that a location server at the user's home site either knows where the user resides, knows how to locate the user, or at the very least knows another location server that possibly might have a better idea. How these servers get this information is outside the scope of SIP itself, but it is expected that many different user-location services will exist for some time. SIP is designed so that it does not care which location service SIP servers actually employ.

8.1. Proxy servers: Relaying and Redirection

If a proxy server receives a request for a user whose location it does not know, and for whom it has no better idea where the user might be, then the server should return a "601 Not Currently Here" reply message.

If the server does have an idea how to contact the user, it can either forward (relay) the request itself, or can redirect the invitation initiator to another client that is more likely to know by sending a 603, 301 or 302 response as appropriate. It can also gateway the request into some other form if some other invitation protocol is in use in a region containing the invited user, though in doing so the server is likely to give up being stateless.

Whether to relay the request or to redirect the request is up to the server itself. For example, if the server is on a firewall machine, then it will probably have to relay the request to servers inside the firewall. Additionally, if a local multicast group is to be used for user location, then the server is likely to relay the request. However, if the user is currently away from home, relaying the request makes little sense, and the server is more likely (though not compelled) to send a redirect reply. SIP is policy-free on this issue. In general, local searches are likely to be better performed by relaying whereas wide-area searches are likely to be better performed by redirection.

When SIP uses UDP transport, clients and servers can make multiple simultaneous requests to locate a particular user at low cost. This greatly speeds up any search for the user, and in most cases will only result in one successful response. Although several simultaneous paths may reach the same host, successful responses arriving from multiple paths will not confuse the client as they should all contain the same successful host address. However, this does imply that paths with many levels of relaying should be strongly discouraged as if the request is fanned out at each hop and relayed many times, request implosions could result. Thus *servers that are not the first hop servers in a chain of servers SHOULD NOT make multiple parallel requests*, but should send a redirection response with multiple alternatives. Thus a firewall host can still perform a parallel search but can control the fanout of the search.

8.2. Parallel Searches: Initiator Behaviour

The session inititor may make a parallel search for a user. This can occur when DNS resolution results in multiple addresses, or when contacting a remote server results in a "603 Alternative Address" response containing multiple addresses to try. All such parallel searches for the same SIP request MUST contain the same SIP Id, though the sequence number (given in the "Path:" field) SHOULD be different for each of the parallel searches.

Whilst performing a parallel search, different responses may result from different servers, and it is important for the initiating client to handle these responses correctly. In general, the following rules apply:

- If a 2xx response is received, the invitation was successful, the user should be informed and all pending requests should be terminated and/or ignored.
- If a 4xx response is received the invitation has definitively failed, the user should be informed, and all pending requests should be terminated and/or ignored.
- If a 3xx response is received, the search should be terminated and all pending requests should be terminated and/or ignored. However, further action MAY be taken depending on the actual reply without informing the user or alternatively the invitation MAY be regarded as having failed in which case the user MUST be informed.
- If a 5xx or 6xx response is received, the particular server responding is removed from the parallel search and the search continues. If a "603 Alternative Address" response is received, the search may be expanded to include those servers listed in the response that have not already responded. The user SHOULD NOT be informed unless there are no other servers left to try, in which case the user MUST be informed.
- If a 1xx response is received, the search continues. The user MAY be informed as deemed appropriate.

8.3. Parallel Searches: Proxy Behaviour

In the same way that an Initiating Client can discover multiple addresses to try, a proxy server can also discover multiple addresses that it may try. For a proxy server to be stateless, it must not make multiple SIP requests because it would then be possible to return a 5xx or 6xx response to the Initiating Client and afterwards obtain a definitive answer. To be able to make multiple parallel SIP requests, it must keep state as to the replies it has already received and MUST NOT return any reply other than 1xx informational replies until it has received a definitive reply or has no further addresses to try.

Thus faced with DNS resolution giving multiple addresses, a proxy server that wishes to be stateless should only send a SIP request to the first address. Similarly a stateless proxy should not attempt to send SIP request to multiple addresses given in a ‘‘603 Alternative Address’’ response that is returned it it, but should forward such a response back towards the initiator.

Proxies that wish to keep state should follow the following rules regarding responses obtained during a parallel search:

- If a 2xx response is received, the invitation was successful, the 2xx response should be forwarded back towards the initiator, and all pending requests should be terminated and/or ignored.
- If a 4xx response is received the invitation has definitively failed, the 4xx response should be forwarded back towards the initiator, and all pending requests should be terminated and/or ignored.
- If a 3xx response is received the invitation is regarded by the proxy as having failed, the 3xx response should be forwarded back towards the initiator, the search should be terminated and all pending requests should be terminated and/or ignored.
- If a 5xx or 6xx response is received, the particular server responding is removed from the parallel search and the search continues. If a ‘‘603 Alternative Address’’ response is received, the search may be expanded to include those servers listed in the response that have not already responded. No response other than a periodic ‘‘100 Trying’’ resonse should be send towards the initiator unless there are no other servers left to try, in which case a response SHOULD be sent as described below.
- If a 1xx response is received, the search continues. The 1xx response MAY be forwarded towards the initiator as appropriate.

If a proxy had exhausted its search and still not obtained a definitive response (it received only 1xx, 5xx, and 6xx responses) the proxy should cache these responses and return the first response from the following ordered list:

1. - 503 Service Unavailable
2. - 500 Server Internal Error
3. - 501 Not Implemented
4. - any other 5xx error not yet defined
5. - 600 Search Failure
6. - 602 Not Currently Here
7. - 601 Not Known Here
8. - any other 6xx error response not yet defined

If a proxy has exhausted its search and the only response it has received has been ‘‘603 Alternative Address’’, then the proxy should send a ‘‘600 Search Failure’’ response if any connection attempt timed out or failed, or it should send ‘‘602 Not Currently Here’’ if two or more ‘‘603 Alternative Address’’ responses only provide references to each other.

8.4. Change of Transport at a Proxy

Editors note: this section is still incomplete. Several options exist for where the responsibilty should lie for retransmissions from proxies between TCP and UDP transport. This section generally assumes local retransmission, but end-to-end transmission through a chain of proxies is also possible

It is possible that a proxy server will receiver a request using TCP and relay it onwards using UDP or vice-versa. SIP does not assume end-to-end reliability even when the initiating client is using TCP, but a SIP client sending a request over TCP MAY assume that the request has been received by the server it sent the request to. Retransmission of the request is then not the responsibility of the client. However, a called user agent SHOULD NOT assume that a 2xx success response has been received by the invitation initiator, even if all the path fields in the request indicated TCP transport because it cannot be certain all those TCP connections still exist. If the called user agent requires knowledge that the response did reach the invitation initiator, it MAY add a ``Confirm:required`` field to the reply as it would if the response was sent using UDP.

In the following, the term ``TCP->UDP proxy'' is used to mean a proxy that received a *request* using TCP and relayed it using UDP. Similarly a ``TCP->UDP proxy'' receives a *reply* using UDP and should relay it using TCP.

8.4.1. Retransmission from a TCP->UDP Proxy

A proxy receiving a request with TCP transport and forwarding that request using UDP becomes responsible for restransmission of the request as required and for timing out the request if no answer is forthcoming.

8.4.2. Retransmissions arriving at a UDP->TCP Proxy

A proxy receiving a request using UDP transport and forwarding that request using TCP transport may have have SIP *request* state associated with that TCP connection or SIP *response* state accociated with it.

If such a proxy receives a retransmission of the UDP request whilst in the state or awaiting a response (i.e, has *request* state), it SHOULD NOT forward the duplicate request into the TCP connection unless the request has been modified, but instead SHOULD respond with a ``100 Trying'' response sent back towards the initiator.

Note: this behaviour is different from a UDP->UDP proxy which MUST forward the retransmitted request and MAY additionally respond with a "100 Trying" response sent back towards the initiator.

If such a proxy receives a retransmission of the UDP request in *response* state (i.e, it has already sent a definitive response) then the proxy MAY retransmit that response if it has cached it. Alternatively if it has not cached the response, it MAY re-send the request towards the called user agent, either via an existing TCP connection if there is one or via a new TCP connection if there is not, to obtain a retransmission of the response. In the latter case, the proxy MAY additionally respond with a ``100 Trying'' response sent back towards the initiator.

Note: this behaviour is the same as a UDP->UDP proxy in the same circumstances.

8.4.3. Confirmation arriving at a TCP->UDP Proxy

One possible event that may occur is that whilst performing a search using UDP, a response may arrive that should be relayed back towards the initiator using TCP, but the TCP connection has been terminated by the initiator. In this case the proxy MUST NOT attempt to relay the response (by opening a TCP connection) and should terminate any outstanding search. In this circumstance only, if the response was a "200 OK" response with a "`Confirm:required`" field, the proxy MAY re-send the request to the Contact Host with a "`Confirm:false`" field to speed hang-up discovery at the called user agent.

8.4.4. Confirmation sent from a UDP->TCP Proxy

Normally a response that arrives at a proxy using TCP that should be sent back towards the initiator using UDP should be sent once, and should only be re-sent if the request is re-sent from the UDP proxy closer to the initiator. However, this does not allow for reliable confirmation.

9. Security Considerations

TBD

Acknowledgments

We wish to thank the members of the IETF MMUSIC WG for their comments and suggestions.

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, MY 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] M. Handley, V. Jacobson ''SDP: Session Description Protocol'' Internet Draft draft-ietf-mmusic-sdp-03.txt, Work in Progress, Nov 1996.

[2] M. Handley, ''SAP: Session Announcement Protocol'' Internet Draft draft-ietf-mmusic-sap-01.txt, Work in Progress, Nov 1996.

[3] Schooler, E.M., ''Case Study: Multimedia Conference Control in a Packet-switched Teleconferencing System'' Journal of Internetworking: Research and Experience, Vol.4, No.2, pp.99-120, June 1993; also available as an ISI technical report ISI/RS-93-359, Aug 1993. ftp://ftp.isi.edu/pub/hpcc-papers/mmc/joi.ps

[4] A. Gulbrandsen, P. Vixie, ''A DNS RR for specifying the location of services'' Internet Draft draft-gulbrandsen-dns-rr-srvcs-02.txt, Work in Progress, Jan 1996.