Mark Handley/Eve Schooler
UCL/Caltech
22 Feb 1996

Session Invitation 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 Invitation 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.

Status of this Memo

1 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 Directory Announcement Protocol (SDAP), provide a mechanism for the former [1] [2]. This document presents a strawman protocol, the Session Invitation Protocol (SIP), to perform the latter. SIP also relies on SDP to specify what is meant by a session.

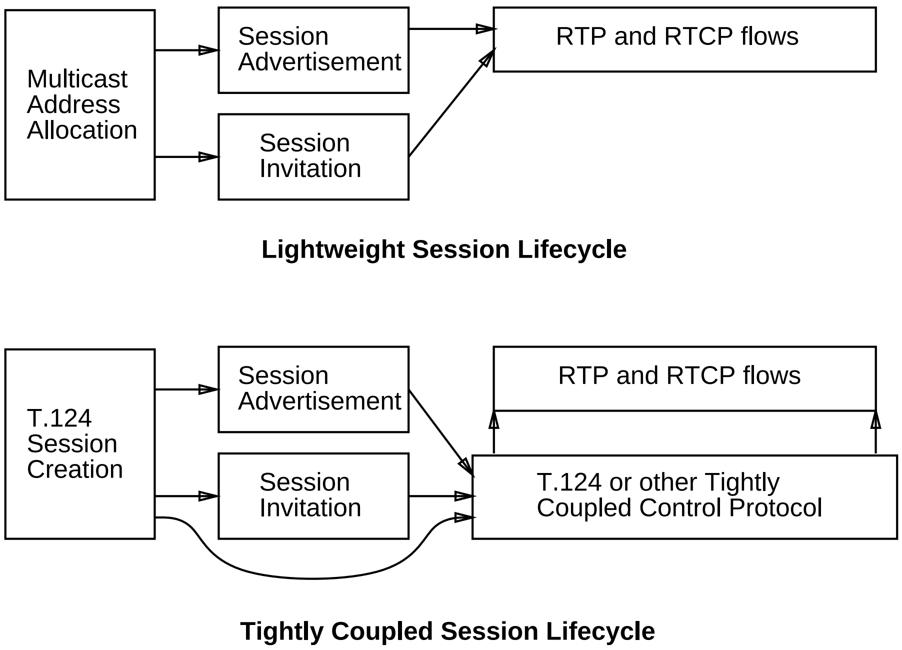

Figure 1: Internet multimedia conferencing lifecycles

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 Invitation Protocol and the Session Directory Announcement Protocol may be invoked or re-invoked at later stages in a session's lifecycle.

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 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 (`mhandley`) with a machine host name (`chopin.cs.ucl.ac.uk`) 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 (`mhandley@chopin.cs.ucl.ac.uk`). 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.

3 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 [5] 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 the user address of the invitee is found, the second and subsequent phases basically implement a request-response protocol. A session description (using SDP format) is sent to the contact address with an invitation for the user to join the session. This request is a single UDP datagram (the format of which 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 following responses:

Category 1: Request Successful

`SUCCESS` - the request was successful in contacting the user, and the user has agreed to participate.

Category 2: Request Unsuccessful

`UNSUCCESSFUL` - the user agent or proxy server knows the user's address, but the request was unsuccessful in contacting the user, for instance if the particular client is not running at that address.

`BUSY` - the user's machine was successfully contacted but the user is busy, or the user does not wish to participate.

`DECLINE` - the user's machine was successfully contacted but the user explicitly does not wish to participate.

`UNKNOWN` - the user is not known.

`FAILED` - the call was unsuccessful due to an unspecified reason.

`FORBIDDEN` - the call was rejected due to an authorisation failure.

Category 3: Progress Reports

`RINGING` - the user agent or conference server is ringing the user.

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

Category 4: Further Action Required

`REDIRECT` - the requesting client should retry on the new address(es) given.

`ALTERNATIVE` - the call was not successful, but alternative services are possible. The alternative services are described in the body of the reply.

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

The initiating client should expect to receive one of these replies within a short time interval. If it does not, it should retry periodically. As a user is usually involved in the feedback loop, a retransmission interval and maximum number of retries is recommended, though we consider these values application-specific. If the number of timeouts is exceeded, the invitation request is considered to have `FAILED`.

SIP does not mandate that any particular retransmission interval be used, though an initial retransmission interval of one second with an exponential back-off if no reply is received is considered appropriate for most implementation purposes.

Responses to the invitation request are not normally acknowledged by the invitation initiator, with the exception of some `SUCCESS` replies (see Section 3.2). If a client receives a `SUCCESS` reply, then there is no need to take any further action.

If a user agent or server does not wish to give hints at whether a user exists or is authorised to talk to the requesting user, then a `FAILED` response may be sent instead of `UNKNOWN`, `BUSY`, `DECLINE`, `FORBIDDEN` or `UNSUCCESSFUL` responses, though for normal usage this is discouraged.

If the client receives a `RINGING` or a `TRYING` reply, it should retry periodically until a concrete reply has been received indicating `SUCCESS`, an unsuccessful request, or that further action is required. Otherwise, the remote user agent or server may time out the connection attempt after a short while. `RINGING` or `TRYING` responses received after a `SUCCESS` response should be ignored.

If the client receives a `REDIRECT` response, it should retry connecting to the address(es) given. If multiple addresses are given, it is up to the application whether to retry connecting to some or all of the addresses simultaneously or sequentially.

If the client receives an `ALTERNATIVE` response, then it is up to the client what course of action to take next depending on the available alternatives. These alternatives may include answering services, alternative users who may be able to answer the call on the user's behalf, or other services not yet envisaged.

If the user cannot support the call, its agent or server can respond with a `NEGOTIATE` response. This should give alternative session options. The client can then decide whether to send a follow-up request, or to give up. The user being called will not usually be contacted until the request contains an acceptable set of encodings. In other words, the user agent or server will only ever forward viable invitations to the user.

3.1 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 either case, if a request was unicast, then the reply is unicast back to the requester; if the request was multicast, the reply is multicast to the same group to which the request was sent.

In all cases where a request is forwarded, each host relaying the message adds 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 are 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.

3.2 Reliability

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 progress report, the initiating client should still continue retransmitting the request, albeit less frequently.

When the remote user agent or server sends a final response (not a progress report), it cannot be sure the client has received the response, and thus should (if the cost of obtaining the results again is sufficiently high) cache the results until a connection setup timeout has occurred.

This means that a user can be contacted successfully, but that the reply that the user was 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 `SUCCESS` reply. Although not required, in the case of a successful invitation the invited user's agent can make a reverse acknowledgment request in its `SUCCESS` reply. In this case the initiator's agent sends a single request with a reply `ACK` if the request was still valid or a reply `NACK` if it was not so that a premature hang-up can be detected without a long timeout. Such a reverse `ACK` request may be retransmitted by the invited user's agent if it so desired. Reverse `ACK` requests can only be made with `SUCCESS` replies, and only the invitation initiator's agent may issue the acknowledgment.

Only a `SUCCESS` reply warrants such an acknowledgment 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, no state is maintained. In particular, when a server makes multiple proxy requests, failure replies do not immediately get passed back to the invitation initiator, and so no end-to-end acknowledgment of a failed request is possible.

4 The User Agent Proxy: Relaying or Redirection

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.

If such 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 an `UNKNOWN` reply message.

If the server does have an idea how to contact the user, it can either forward the request itself, or can redirect the invitation initiator to another client that is more likely to know. It can also gateway the request into some other form if some other invitation protocol is in use in a region that it likely to contact 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 practice however, local searches are likely to be better performed by relaying whereas wide-area searches are likely to be better performed by redirection.

As SIP is UDP based, clients and servers can make multiple simultaneous requests to locate a particular user. 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, both request and response implosions could result. Thus *servers that are not the first hop servers in a chain of servers should not make multiple 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.

5 Message Formats

All messages are text-based, and each is sent in a single UDP datagram.

5.1 Session Invitations

An example of a session invitation is shown below. It is divided into 5 sections:

- The path field (PA) 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.
- The authentication field (AU) provides a digital signature of the remaining fields.
- The request identifier (ID) gives a globally unique identifier for the invitation.
- The request header consists of both a from field (FR), indicating the invitation initiator, and a to field (TO), specifying the invited user.
- The session description gives details of the session the user is being invited to join.

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

```
SIP/1.0 REQ
PA=128.16.64.19 128.16.5.31 239.128.16.254/16
AU=none
ID=128.16.64.19/32492374
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
v=0
o=van 2353655765 2353687637 IN IP4 128.3.4.5
s=Mbone Audio
i=Discussion of Mbone Engineering Issues
e=van@ee.lbl.gov (Van Jacobsen)
c=IN IP4 224.2.0.1/127
t=0 0
m=audio 3456 RTP PCMU
```

The first line above states that this is a SIP version 1.0 request.

The path field (`PA=`...) gives 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.

The full authentication field format is not yet defined, and the above message contains the entry `AU=none`, meaning no authentication is included.

The request header above states that the request was initiated by `M.Handley@cs.ucl.ac.uk` (specifically it was initiated from 128.16.64.19, as can be seen from the path field) and the user being invited is `J.Crowcroft@cs.ucl.ac.uk`.

The session description is a Session Description Protocol (SDP) description as defined in the companion draft[1]. 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.

5.2 Normal Replies

An example reply is given below. The first line of the reply states the SIP version number, that it's a category 1 reply, and the specific type of category 1 reply: `SUCCESS`. The path field is taken from the request, and entries in it 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 `FR` and `TO` fields is preserved. In addition, a contacted host entry 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. This contacted host field may be encrypted using the invitation initiators public key if required, in which case four random bytes will be added to the start of the CH field before encryption.

```
SIP/1.0 REP 1 SUCCESS
PA=128.16.64.19 128.16.5.31 239.128.16.254/16
AU=none
ID=128.16.64.19/32492374
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
CH=IN IP4 128.16.8.75
```

This same format is used from replies from categories 1, 2 and 3. Category 4 replies require addition information to be added.

If the invited user's agent requires confirmation of receipt of a SUCCESS reply, it may optionally add an additional reliability mode RM field to the body of the message specifying that an ACK is required. This is only permitted with category 1 replies. An example is:

```
SIP/1.0 REP 1 SUCCESS
PA=128.16.64.19 128.16.5.31 239.128.16.254/16
AU=none
ID=128.16.64.19/32492374
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
CH=IN IP4 128.16.8.75
RM=ACK
```

In response to such a request, the invitation initiators agent should retransmit its request with either an ACK or a NACK flag stating whether the session still exists or no longer exists respectively (see section 3.2 for details). An example of an ACK request is:

```
SIP/1.0 REQ ACK
PA=128.16.64.19 128.16.5.31 239.128.16.254/16
AU=none
ID=128.16.64.19/32492374
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
v=0
o=van 2353655765 2353687637 IN IP4 128.3.4.5
s=Mbone Audio
i=Discussion of Mbone Engineering Issues
```

```
e=van@ee.lbl.gov (Van Jacobsen)
c=IN IP4 224.2.0.1/127
t=0 0
m=audio 3456 RTP PCMU
```

5.3 Category 4 "Further Action Required" Replies

Redirects

An example of a redirect reply is:

```
SIP/1.0 REP 4 REDIRECT
PA=128.16.64.19 128.16.5.31
AU=none
ID=128.16.64.19/32492374
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
CH=IN IP4 128.16.5.31
RE=IN IP4 239.128.16.254/16
```

In this example, the client is suggested 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 client is within the scope of the local group.

Alternative

An example of an ALTERNATIVE reply is:

```
SIP/1.0 REP 4 ALTERNATIVE
PA=128.16.64.19 128.16.5.31
AU=none
ID=128.16.64.19/32492374
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
CH=IN IP4 128.16.5.31
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
```

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 (128.16.5.31) with the session description provided. This request should contain a different session id from the one in the original request. An example would be:

```
SIP/1.0 REQ
PA=128.16.64.19
AU=none
ID=128.16.64.19/32492375
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
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 an `ALTERNATIVE` reply as a failure if they wish to do so.

Negotiate

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

- Insufficient Bandwidth - the bandwidth specified in the session description or defined by the media exceeds that known to be available.
- Incompatible Media - a media format described in the request is not available.
- Multicast not available - the site where the user is located does not support multicast.
- 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 `NEGOTIATE` reply.

A complex example of a NEGOTIATE reply is:

```
SIP/1.0 REP 4 NEGOTIATE
PA=128.16.64.19 128.16.5.31
AU=none
ID=128.16.64.19/32492376
FR=M.Handley@cs.ucl.ac.uk
TO=J.Crowcroft@cs.ucl.ac.uk
CH=IN IP4 128.16.5.31
NO=BANDWIDTH
b=CT:256
AB=CT:128
NO=MEDIA
m=audio 3456 RTP GSM
AM=RTP DVI
AM=RTP PCM
AM=RTP LPC
NO=MEDIA
m=video 3458 RTP NV
AM=RTP H261
NO=MEDIA
m=whiteboard 3460 UDP WB
NO=MULTICAST
```

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, NV video, and WB whiteboard. None of these are available, but the reply states that DVI, PCM or LPC audio could be supported in order of preference and that H261 video could be supported, although no whiteboard can be supported. The reply also states that multicast is not available.

Invitation initiators can choose to treat NEGOTIATE replies as a failure if they wish to do so.

5.4 Unknown Replies

It is expected that future versions of SIP will need to add new reply types. If these are in categories 1-4 the default behaviour of old clients should be:

Category 1 : Treat as SUCCESS and send an ACK or NACK as appropriate.

Category 2 : Treat as FAIL.

Category 3 : Treat as TRYING.

Category 4 : Treat as FAIL

Replies to a SIP/1.0 request should not be in any other category.

6 Protocol Syntax

```
SIP-Request ::=          RequestType
                         PathHeader
                         AuthHeader
                         RequestId
                         RequestHeader
                         SessionDescriptor


RequestType ::=          VersionId space "REQ"
                         [space ("ACK" | "NACK")] newline


VersionId ::=            "SIP/1.0"


PathHeader ::=           "PA=" Address (space Address)* newline;


AuthHeader ::=           "AU=" [ "none" | AuthField ] newline;


AuthField ::=            ;To Be Defined...


RequestId ::=            String


RequestHeader ::=        "FR=" UserAddress newline "TO=" UserAddress newline


SessionDescriptor ::=    Announcement
                         ;defined in SDP specification


SIP-Reply ::=            ReplyType
                         PathHeader
                         AuthHeader
                         RequestId
                         RequestHeader
                         ReplyBody
```

```
ReplyType ::=            VersionId space "REP" space Category
                         space ReplyTypeName newline

Category ::=             "1" | "2" | "3" | "4"

ReplyTypeName ::=        "SUCCESS" | "UNSUCCESSFUL" | "BUSY" |
                         "DECLINE" | "UNKNOWN" | "FAILED" |
                         "FORBIDDEN" | "RINGING" | "TRYING" |
                         "REDIRECT" | "ALTERNATIVE" | "NEGOTIATE"

ReplyBody ::=            ContactHost newline
                         [ RedirectBody | AlternativeBody | NegotiateBody]

ContactHost ::=          NetType space AddrType space UniAddress newline

RedirectBody ::=         NetType space AddrType space Address newline

AlternativeBody ::=      SessionDescription

NegotiateBody ::=        (Reason)+

Reason ::=               NoBandwidth | NoMedia | NoMulticast | NoUnicast

NoBandwidth ::=          "NO=BANDWIDTH" newline
                         SDP-BandWidthField SIP-BandWidthField

SDP-BandWidthField ::=   "b=" bwtype ":" bandwidth newline
                         ;bwtype and bandwidth are define in the SDP spec

SIP-BandWidthField ::=   "AB=" bwtype ":" bandwidth newline
                         ;bwtype and bandwidth are define in the SDP spec

NoMedia ::=              "NO=MEDIA" newline
                         SDP-MediaField ( AvailMedia )+

SDP-MediaField ::=       "m=" media space port space proto space fmt newline
                         ;media, port, proto, and fmt are defined in the
                         ;SDP spec
```

```
AvailMedia ::=          "AM=" proto space fmt newline


NoMulticast ::=         "NO=MULTICAST" newline


NoMulticast ::=         "NO=UNICAST" newline


Address ::=             UniAddress | MultiAddress "/" TTL


UniAddress ::=          IP4-address | IP6-address


IP4-address ::=         b1 ``.'' decimal_uchar ``.'' decimal_uchar ``.'' b4
b1 ::=                  decimal_uchar
                        ;less than ``224''; not ``0'' or ``127''
b4 ::=                  decimal_uchar
                        ;not ``0''


IP6-address ::=         ;to be defined


TTL ::=                 decimal_uchar


decimal_uchar ::=       DIGIT
                        | (POS-DIGIT DIGIT)
                        | ("1" 2*DIGIT)
                        | ("2" ("0"|"1"|"2"|"3"|"4") DIGIT)
                        | ("2" "5" ("0"|"1"|"2"|"3"|"4"|"5"))


DIGIT ::=               "0" | POS-DIGIT


POS-DIGIT ::=           "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"


ALPHA ::=               a | b | c | d | e | f | g | h | i | j | k |
                        l | m | n | o | p | q | r | s | t | u | v |
                        w | x | y | z | A | B | C | D | E | F | G |
                        H | I | J | K | L | M | N | O | P | Q | R |
                        S | T | U | V | W | X | Y | Z
```

```
unicode-safe ::=        ALPHA | DIGIT |
                        ""' | "(" | ")" | ""' | "-" | "." | "/" | ":" |
                        "?" | """ | "#" | "$" | "&" | "*" | ";" | "<" |
                        "=" | ">" | "@" | "[" | "]" | "^" | "_" | "`" |
                        "{" | "|" | "}" | "+" | space | tab
                        ;although unicode allows newline and carriage
                        ;return, we don't here.


space ::= ;ascii code 32


newline ::= ;ascii code 10
```

7 Acknowledgments

This work is based on the work of many others, and in particular on the MMCC work by Eve Schooler[4], on the Personal Mobility work by Henning Schulzrinne[3], and on the IETF MMUSIC group's Session Description Protocol[1].

Authors' Addresses

Mark Handley
Department of Computer Science
University College London
London WC1E 6BT
United Kingdom
electronic mail: M.Handley@cs.ucl.ac.uk

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-02.txt, Work in Progress, Feb 1996.

[2] M. Handley, “SDAP: Session Directory Announcement Protocol” Internet Draft draft-ietf-mmusic-sdap-00.txt, Work in Progress, Feb 1996.

[3] Schulzrinne, H., “Personal Mobility for Multimedia Services in the Internet” IMDS’96, March 4-6 1996. ftp://ftp.fokus.gmd.de/pub/step/papers/Schu9603:Personal.ps.gz

[4] 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

[5] 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.