Internet Engineering Task Force **MMUSIC WG**
INTERNET-DRAFT
draft-ietf-mmusic-sdp-02.ps
Mark Handley/Van Jacobson
ISI/LBNL
21st Nov 1996
Expires: 21st May 1997

SDP: Session Description Protocol

Status of this Memo

Abstract

This document defined the Session Description Protocol, SDP. SDP is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of session initiation.

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. Introduction

On the Internet multicast backbone (Mbone), a session directory tool is used to advertise multimedia conferences and communicate the conference addresses and conference tool-specific information necessary for participation. This document defines a session description protocol for this purpose, and for general real-time multimedia session description purposes. This draft does not describe multicast address allocation or the distribution of SDP messages in detail. These are described in accompanying drafts.

2. Background

The Mbone is the part of the internet that supports IP multicast, and thus permits efficient many-to-many communication. It is used extensively for multimedia conferencing. Such conferences usually have the property that tight coordination of conference membership is not necessary; to receive a conference, a user at an Mbone site only has to know the conference's multicast group address and the UDP ports for the conference data streams.

Session directories assist the advertisement of conference sessions and communicate the relevant conference setup information to prospective participants.

This document defines a session description protocol that is used to define multimedia sessions. It is also an attempt to provide guidelines to the writers of such announcement tools to protect the Mbone from misuse and to preserve the inherent scalability of the original LBNL sd session directory program whilst enhancing its functionality.

We have also attempted to enhance the generality of SDP so that it can be used for a wider range of network environments and applications than just multicast session directories.

3. Glossary of Terms

The following terms are used in this document, and have specific meaning within the context of this document.

Conference
: A multimedia conference is a set of two or more communicating users along with the software they are using to communicate.

Session
: A multimedia session is a set a multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session.

Session Advertisement
: See session announcement.

Session Announcement
: A session announcement is a mechanism by which a session description is conveyed to users is a pro-active fashion, i.e., the session description was not explicitly requested by the user.

Session Description
: A well defined format for conveying sufficient information to discover and participate in a multimedia session.

4. SDP Usage

4.1. Multicast Announcements

SDP is a session description protocol for multimedia sessions. Normally it is used by an SDP client that announces a conference session by periodically multicasting an announcement packet on a well known multicast address and port. With the advent of administrative scoping in the Mbone, it is likely that session directory clients will need to be able to listen for such announcements on multiple addresses. This multicasting of announcements is performed by the Session Announcement Protocol (SAP).

SAP packets are UDP packets with the following format:

```
0                    31
|--------------------|
| SAP header         |
|--------------------|
| text payload       |
|/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\|
```

The header is the Session Announcement Protocol header. SAP is described in more detail in a companion draft [4]

The text payload is an SDP session description, as described in this draft. The text payload should be no greater than 1 Kbyte in length. If announced by SAP, only one session announcement is permitted in a single packet.

4.2. Email and WWW Announcements

For both email and WWW distribution, the use of the MIME content type "`application/x-sdp`" is currently suggested. This enables the automatic launching of applications from the WWW client or mail reader in a standard manner.

Note that announcements of multicast sessions made only via email or the World Wide Web (WWW) do not have the property that the receiver of a session announcement can necessarily receive the session, nor do they allow scalable dynamic multicast address allocation.

5. Requirements and Recommendations

The purpose of SDP is to convey information about media streams in multimedia sessions to allow the recipients of a session description to participate in the session. SDP is primarily intended for use in an internetwork, although it is sufficiently general that it can describe conferences in other network environments.

A multimedia session, for these purposes, is defined as a set of media streams that exist for some duration of time. Media streams can be many-to-many. The times during which the session is active need not be continuous.

Thus far, multicast based sessions on the internet have differed from many other forms of conferencing in that anyone receiving the traffic can join the session (unless the session traffic is encrypted). In such an environment, SDP serves *two* primary purposes. It is a means to communicate the existence of a session, and is a means to convey sufficient information to enable joining and participating in the session. In a unicast environment, only the latter purpose is likely to be relevant.

Thus SDP includes:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session

- Information to receive those media (addresses, ports, formats and so on)

As resources necessary to participate in a session may be limited, some additional information may also be desirable:

- Information about the bandwidth to be used by the conference
- Contact information for the person responsible for the session

In general, SDP must convey sufficient information to be able to join a session (with the possible exception of encryption keys) and to announce the resources to be used to non-participants that may need to know.

5.1. Media Information

SDP includes:

- The type of media (video, audio, etc)
- The transport protocol (RTP/UDP/IP, H.320, etc)
- The format of the media (H.261 video, MPEG video, etc)

For an IP multicast session, the following are also conveyed:

- Multicast address for media
- Transport Port for media

For an IP unicast session, the following are conveyed:

- Contact address for media
- Transport port for contact address

This may or may not be be the source and destination of the media stream.

Sessions being conveyed over other types of network will have their own specific requirements - SDP must be extensible for these.

5.2. Timing Information

Sessions may either be bounded or unbounded in time. Whether or not they are bounded, they may be only active at specific times.

SDP can convey:

- An arbitrary list of start and stop times bounding the session
- For each bound, repeat times such as "every Wednesday at 10am for one hour"

This timing information is globally consistent, irrespective of local time zone or daylight saving time

5.3. Private Sessions

It is possible to create both public sessions and private sessions. Private sessions will typically be conveyed by encrypting the session description to distribute it. The details of how encryption is performed are dependent on the mechanism used to convey SDP - see [4] for how this is done for session announcements.

If a session announcement is private it is possible to use that private announcement to convey encryption keys necessary to decode each of the media in a conference, including enough information to know which encryption scheme is used for each media.

5.4. Further Information

SDP should convey enough information to decide whether or not to participate in a session. It should include additional pointers in the form of Universal Resources Identifiers (URIs) for more information about the session.

5.5. Categorisation

When many session descriptions are being distributed by SAP or any other advertisement mechanism, it may be desirable to filter announcements that are of interest from those that are not. SDP should support a categorisation mechanism for sessions that is capable of being automated.

5.6. Internationalization

The SDP specification recommends the use of 8-bit ISO 8859-1 character sets to allow the extended ASCII characters used by many western and northern European languages to be represented. However, there are many languages that cannot be represented in an ISO 8859-1 character set. SDP should also allow extensions to allow other font types to be used when required.

6. SDP Specification

SDP session descriptions are entirely textual. The textual form, as opposed to a binary encoding such as ASN/1 or XDR, was chosen to enhance portability, to enable a variety of transports to be used (e.g, session description in a MIME email message) and to allow flexible, text-based toolkits (e.g., Tcl/Tk) to be used to generate and to process session descriptions. However, since the total bandwidth allocated to all SAP announcements is strictly limited, the encoding is deliberately compact. Also, since announcements may be transported via very unreliable means (e.g., email) or damaged by an intermediate caching server, the encoding was designed with strict order and formatting rules so that most errors would result in malformed announcements which could be detected easily and discarded. This also allows rapid discarding of encrypted announcements for which a receiver does not have the correct key.

An SDP session description consists of a number of lines of text of the form
<type>=<value>
<type> is always exactly one character and is case-significant. *<value>* is a structured text string whose format depends on *<type>*. Whitespace is not permitted either side of the '=' sign. In general *<value>* is either a number of fields delimited by a single space character or a free format string.

Each announcement consists of a session description section followed by zero or more 'media' description sections. The session description starts with a 'v=' line and continues to the first media description or the next session description. The media description starts with an 'm=' line and continues to the next media description or session description. When SDP is conveyed by SAP, only one session description is allowed per packet. When SDP is conveyed by other means, many SDP session descriptions may be concatenated together. Some lines in each description are required and some are optional but all must appear in exactly the order given here (the fixed order greatly enhances error detection and allows for a simple parser). Optional items are marked with a ‘*’.

```
Session description
        v=  (protocol version)
        o=  (owner/creator and session identifier).
        s=  (session name)
        i=* (session information)
        u=* (URI of description)
        e=* (email address)
        p=* (phone number)
        c=* (connection information - not required if included in all media)
        b=* (bandwidth information)
      One or more time descriptions
        z=* (time zone adjustments)
        k=* (encryption key)
        a=* (zero or more session attribute lines)
      Zero or more media descriptions

Time description
        t=  (time the session is active)
        r=* (zero or more repeat times)

Media description
        m=  (media name and transport address)
        i=* (media title)
        c=* (connection information - optional if included at session-level)
        b=* (bandwidth information)
        k=* (encryption key)
        a=* (zero or more media attribute lines)
```

The set of ‘type’ letters is deliberately small and not intended to be extensible -- SDP parsers must completely ignore any announcement that contains a ‘type’ letter that it does not understand. The ‘attribute’ mechanism (described below) is the primary means for extending SDP and tailoring it to particular applications or media. Some attributes (the ones listed in this document) have a defined meaning but others may be added on an application-, media- or session-specific basis. A session directory must ignore any attribute it doesn’t understand.

The connection (‘c=’) and attribute (‘a=’) information in the session section applies to all the media of that session unless overridden by connection information or an attribute of the same name in the media description. For instance, in the example below, each media behaves as if it were given a ‘`recvonly`’ attribute.

An example SDP description is:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 3456 RTP/AVP 0
m=video 2232 RTP/AVP 31
m=whiteboard 32416 UDP WB
a=orient:portrait
```

Text records such as the session name and information may contain any printable 8 bit ISO 8859-1 character with the exceptions of 0x0a (newline) and 0x0d (carriage return). Carriage Return is prohibited, and Newline is used to end a record.

Protocol Version

`v=0`

The "v" field gives the version of the Session Description Protocol. There is no minor version number.

Origin

`o=`*<username> <session id> <version> <network type> <address type> <address>*

The "o" field gives the originator of the session (their username and the address of the user's host) plus a session id and session version number. *<username>* is the user's login on the originating host, or it is "-" if the originating host does not support the concept of user ids. *<username>* must not contain spaces. *<session id>* is a numeric string such that the triple of *<username>*, *<session id>* and *<address>* form a globally unique identifier for the session. The method of session id allocation is up to the creating tool, but it has been suggested that a Network Time Protocol (NTP) timestamp be used to ensure uniqueness [1]. *<version>* is a version number for this announcement. It is needed for proxy announcements to detect which of several announcements for the same session is the most recent. Again its usage is up to the creating tool, so long as *<version>* is increased when a modification is made to the session data. Again, it is recommended (but not mandatory) that an NTP timestamp is used. *<network type>* is a text string giving the type of network. Initially "`IN`" is defined to have the meaning "Internet". *<address type>* is a text string giving the type of the address that follows. Initially "`IP4`" and "`IP6`" are defined. *<address>* is the globally unique address of the machine from which the session was created. For an address type of `IP4`, this is the dotted-decimal representation of the IP version 4 address of the machine.

Session Name

`s=`*<session name>*

The "s" field is the session name. There must be one and only one "s" field per announcement, and it must contain printable ISO 8859-1 characters (but see also the '`charset`' attribute below).

Session and Media Information

`i=`*<session description>*

The “i” field is information about the session. There must be no more than one session-level “i” field per session announcement. Although it may be omitted, this is discouraged, and user interfaces for composing sessions should require text to be entered. If it is present it must contain printable ISO 8859-1 characters (but see also the ‘`charset`’ attribute below).

A single “i” field can also be used for each media definition. In media definitions, “i” fields are primarily intended for labeling media streams. As such, they are most likely to be useful when a single session has more than one distinct media stream of the same media type. An example would be two different whiteboards, one for slides and one for feedback and questions.

URI

`u=`*<URI>*

- A URI is a Universal Resource Identifier as used by WWW clients
- The URI should be a pointer to additional information about the conference
- This field is optional, but if it is present it should be specified before the first media field
- No more than one URI field is allowed per session description

Email Address and Phone Number

`e=`*<email address>*
`p=`*<phone number>*

- These specify contact information for the person responsible for the conference. This is not necessarily the same person that created the conference announcement.
- Either an email field or a phone field *must* be specified. Additional email and phone fields are allowed.
- If these are present, they should be specified before the first media field.
- More than one email or phone field can be given for a session description.
- Phone numbers should be given in the conventional international format - preceded by a “+” and the international country code. There must be a space or a hyphen (“-”) between the country code and the rest of the phone number. Spaces and hyphens may be used to split up a phone field to aid readability if desired. For example:

  `p=+44-171-380-7777` or `p=+1 617 253 6011`
- Both email addresses and phone numbers can have an optional free text string associated with them, normally giving the name of the person who may be contacted. This should be enclosed in parenthesis if it is present. For example:

  `e=mjh@isi.edu (Mark Handley)`

  The alternative RFC822 name quoting convention is also allowed for both email addresses and phone numbers. For example,

  `e=Mark Handley <mjh@isi.edu>`

The free text string should be in an IS0-8859-1 character set, or alternatively in unicode UTF-7 encoding if the appropriate `charset` session-level attribute is set.

Connection Data

`c=`*<network type> <address type> <connection address>*

The "c" field contains connection data.

The first sub-field is the network type, which is a text string giving the type of network. Initially "`IN`" is defined to have the meaning "Internet"

The second sub-field is the address type. This allows SDP to be used for sessions that are not IP based. Currently only `IP4` is defined.

The third sub-field is the connection address. Optional extra sub-fields may be added after the connection address depending on the value of the *<address type>* field.

For `IP4` addresses, the connection address is defined as follows:

- Typically the connection address will be a class-D IP multicast group address. If the conference is not multicast, then the connection address contains the unicast IP address of the expected data source or data relay or data sink as determined by additional attribute fields. It is not expected that unicast addresses will be given in a session description that is communicated by a multicast announcement, though this is not prohibited.

- Conferences using an IP multicast connection address must also have a time to live (TTL) value present in addition to the multicast address. The TTL defines the scope with which multicast packets sent in this conference should be sent. TTL values must be in the range 0-255. The Mbone usage guidelines (currently available at ftp://ftp.isi.edu/mbone/faq.txt) define several standard settings for TTL:

```
local net:     1
site:         15
region:       63
world:       127
```

  Other settings may have local meaning (e.g., 31 for all sites within an organization).

  The TTL for the session is appended to the address using a slash as a separator. An example is:

```
c=IN IP4 224.2.1.1/127
```

  Hierarchical or layered encoding schemes are data streams where the encoding from a single media source is split into a number of layers. The receiver can choose the desired quality (and hence bandwidth) by only subscribing to a subset of these layers. Such layered encodings are normally transmitted in multiple multicast groups to allow multicast pruning. This technique keeps unwanted traffic from sites only requiring certain levels of the hierarchy. For applications requiring multiple multicast groups, we allow the following notation to be used for the connection address:

  <base multicast address> / *<ttl>* / *<number of addresses>*

  If the number of addresses is not given it is assumed to be one. Multicast addresses so assigned are contiguously allocated above the base address, so that, for example:

```
c=IN IP4 224.2.1.1/127/3
```

would state that addresses 224.2.1.1, 224.2.1.2 and 224.2.1.3 are to be used at a ttl of 127.

It is illegal for the slash notation described above to be used for IP unicast addresses.

A session announcement must contain one ‘‘c’’ field in each media description (see below) or a ‘‘c’’ field at the session-level. It may contain a session-level ‘‘c’’ field and one additional ‘‘c’’ field per media description, in which case the per-media values override the session-level settings for the relevant media.

Bandwidth

`b=`*<modifier>:<bandwidth-value>*

- This specifies the proposed bandwidth to be used by the session or media, and is optional.
- *<bandwidth-value>* is in kilobits per second
- *<modifier>* is an single alphanumeric word giving the meaning of the bandwidth figure.
- Two modifiers are initially defined:

`CT` *Conference Total*: An implicit maximum bandwidth is associated with each TTL on the Mbone or within a particular multicast administrative scope region (the Mbone bandwidth vs. TTL limits are given in the MBone FAQ). If the bandwidth of a session or media in a session is different from the bandwidth implicit from the scope, a ‘`b=CT:`...’ line should be supplied for the session giving the proposed upper limit to the bandwidth used. The primary purpose of this is to give an approximate idea as to whether two or more conferences can co-exist simultaneously.

`AS` *Application Specific Maximum*: The bandwidth is interpreted to be application specific, i.e., will be the application’s concept of maximum bandwidth. Normally this will coincide with what is set on the application’s ‘‘maximum bandwidth’’ control if applicable.

Note that `CT` gives a total bandwidth figure for all the media at all sites. `AS` gives a bandwidth figure for a single media at a single site, although there may be many sites sending simultaneously.

- **Extension Mechanism:** Tool writers can define experimental bandwidth modifiers by prefixing their modifier with ‘‘`X-`’’. For example:

```
b=X-YZ:128
```

SDP parsers should ignore bandwidth fields with unknown modifiers. Modifiers should be alpha-numeric and, although no length limit is given, they are recommended to be short.

Times, Repeat Times and Time Zones

`t=`*<start time> <stop time>*

- ‘‘t’’ fields specify the start and stop times for a conference session. Multiple ‘‘t’’ fields may be used if a session is active at multiple irregularly spaced times; each additional ‘‘t’’ field specifies an additional period of time for which the session will be active. If the session is active at regular times, an ‘‘r’’ field (see below) should be used in addition to and following a ‘‘t’’ field - in which case the ‘‘t’’ field specifies the start and stop times of the repeat sequence.
- The first and second sub-fields give the start and stop times for the conference respectively. These values are the decimal representation of Network Time Protocol (NTP) time values in seconds [1]. To convert these values to UNIX time, subtract decimal 2208988800.

- If the stop-time is set to zero, then the session is not bounded, though it will not become active until after the start-time. If the start-time is also zero, the session is regarded as permanent.

  User interfaces should strongly discourage the creation of unbounded and permanent sessions as they give no information about when the session is actually going to terminate, and so make scheduling difficult.

  The general assumption may be made, when displaying unbounded sessions that have not timed out to the user, that an unbounded session will only be active until half an hour from the current time or the session start time, whichever is the later. If behaviour other than this is required, an end-time should be given and modified as appropriate when new information becomes available about when the session should really end.

  Permanent sessions may be shown to the user as never being active unless there are associated repeat times which state precisely when the session will be active. In general, permanent sessions should not be created for any session expected to have a duration of less than 2 months, and should be discouraged for sessions expected to have a duration of less than 6 months.

- It is prohibited for the start time to be after the stop time.

`r=`<*repeat interval*> <*active duration*> <*list of offsets from start-time*>

- "r" fields specify repeat times for a session. For example, if a session is active at 10am on Monday and 11am on Tuesday for one hour each week for three months, then the <*start time*> in the corresponding "t" field would be the NTP representation of 10am on the first Monday, the <*repeat interval*> would be 1 week, the <*active duration*> would be 1 hour, and the *offsets* would be zero and 25 hours. The corresponding "t" field stop time would be the NTP representation of the end of the last session three months later. By default all fields are in seconds, so the "r" and "t" fields might be:

  ```
  t=3034423619 3042462419
  r=604800 3600 0 90000
  ```

  To make announcements more compact, times may also be given in units of days, hours or minutes. The syntax for these is a number *immediately* followed by a single case-sensitive character. Fractional units are not allowed - a smaller unit should be used instead. The following unit specification characters are allowed:

  `d` - days (86400 seconds)
  `h` - minutes (3600 seconds)
  `m` - minutes (60 seconds)
  `s` - seconds (allowed for completeness but not recommended)

  Thus, the above announcement could also have been written:

  ```
  r=1d 1h 0 25h
  ```

  Monthly and yearly repeats cannot currently be directly specified with a single SDP repeat time - instead separate "t" fields should be used to explicitly list the session times.

`z=`<*adjustment time*> <*offset*> <*adjustment time*> <*offset*> ....

- Should it be necessary to schedule a repeated session which spans a change from daylight-saving time to standard time or vice-versa, it is necessary to specify offsets from the base repeat times. This is necessary because different time zones change time at different times of day, different countries change to or from daylight time on different dates, and some countries to not have daylight saving time at all.

Thus in order to schedule a session that is at the same time winter and summer, it must be possible to specify unambiguously by whose time zone a session is scheduled. To simplify this task for receivers, we allow the sender to specify the NTP time that a time zone adjustment happens and the offset from the time when the session was first scheduled. The "z" field allows the sender to specify a list of these adjustment times and offsets from the base time.

An example might be:

```
z=2882844526 -1h 2898848070 0
```

This specifies that at time 2882844526 the time base by which the session's repeat times are calculated is shifted back by 1 hour, and that at time 2898848070 the session's original time base is restored. Adjustments are always relative to the specified start time - they are not cumulative.

- If a session is likely to last several years, it is expected that the session announcement will be modified periodically rather than transmit several years worth of adjustments in one announcement.

Encryption Keys

k=<*method*>
k=<*method*>:<*encryption key*>

- In countries where encrypted sessions are not prohibited by law, the session description protocol may be used to convey encryption keys.
- A key field is permitted before the first media entry (in which case it applies to all media in the session), or for each media entry as required.
- The format of keys and their usage is outside the scope of this document, but see [3].

- The *method* indicates the mechanism to be used to obtain a usable key by external means, or from the encoded encryption key given. The following methods are defined:

  k=clear:<*encryption key*>
  : The encryption key (as described in [3] for RTP media streams under the AV profile) is included untransformed in this key field.

  k=base64:<*encoded encryption key*>
  : The encryption key (as described in [3] for RTP media streams under the AV profile) is included in this key field but has been base64 encoded because it includes characters that are prohibited in SDP.

  k=uri:<*URI to obtain key*>
  : A Universal Resource Identifier as used by WWW clients is included in this key field. The URI refers to the data containing the key, and may require additional authentication before the key can be returned. The key should not be obtained until the user wishes to join the session to reduce synchronisation of requests to the WWW server(s).

  k=prompt
  : No key is included in this SDP description, but the session or media stream referred to by this key field is encrypted. The user should be prompted for the key when

attempting to join the session, and this user-supplied key should then used to decrypt the media streams.

Attributes

`a=`*<flag>*
`a=`*<attribute>*`:`*<value>*

A media field may also have any number of attributes (“a” fields) which are media specific. Attribute fields may be of two forms:

- flag attributes. A flag attribute is simply of the form “`a=`*<flag>*”. These are binary attributes, and the presence of the attribute conveys that the attribute is “true”. An example might be “`a=recvonly`”.
- value attributes. A value attribute is of the form “`a=`*<attribute>*`:`*<value>*”. An example might be that a whiteboard could have the value attribute “`a=orient:landscape`”

Attribute interpretation depends on the media tool being invoked. Thus receivers of session descriptions should be configurable in their interpretation of announcements in general and of attributes in particular.

Attribute fields (“a” fields) can also be added before the first media field. These attributes would convey additional information that applies to the conference as a whole rather than to individual media. An example might be the conference’s floor control policy.

Media Announcements

`m=`*<media> <port> <transport> <fmt list>*

A session announcement may contain a number of media announcements. Each media announcement starts with an “m” field, and is terminated by either the next “m” field or by the end of the session announcement. A media field also has several sub-fields:

- The first sub-field is the media type. Currently defined media are “audio”, “video”, “whiteboard”, “text” and “data”, though this list may be extended as new communication modalities emerge (e.g., telepresense or conference control).

- The second sub-field is the transport port to which the media stream will be sent. The meaning of the transport port depends on the network being used as specified in the relevant “`c`” field and on the transport protocol defined in the third sub-field. Other ports used by the media application (such as the RTCP port, see [2]) should be derived algorithmically from the base media port.

  For transports based on UDP, the value should be in the range 1024 to 65535 inclusive. For RTP compliance it should be an even number. If the port is allocated randomly by the creating application, it is recommended that ports above 5000 are chosen as, on Unix systems, ports below 5000 may be allocated automatically by the operating system.

  For applications where hierarchically encoded streams are being sent to a unicast address, it may be necessary to specify multiple transport ports. This is done using a similar notation to that used for IP multicast addresses in the “c” field:

  `m=`*<media> <port>*`/`*<number of ports> <transport> <fmt list>*

  In such a case, the ports used depend on the transport protocol. For RTP, only the even ports are used for data and the corresponding one-higher odd port is used for RTCP. For example:

```
m=video 3456/2 RTP/AVP 31
```

would specify that ports 3456 and 3457 form one RTP/RTCP pair and 3458 and 3459 form the second RTP/RTCP pair. `RTP/AVP` is the transport protocol and 31 is the format (see below).

It is illegal for both multiple addresses to be specified in the ‘‘c’’ field and for multiple ports to be specified in the ‘‘m’’ field in the same session announcement.

- The third sub-field is the transport protocol. The transport protocol values are dependent on the address-type field in the ‘‘c’’ fields. Thus a ‘‘c’’ field of `IP4` defines that the transport protocol runs over IP4. For IP4, it is normally expected that most media traffic will be carried as RTP over UDP. However, some commonly used applications such as vat [5] do not use RTP. Thus the following transport protocols are defined:

  - `RTP/AVP` - the IETF’s Realtime Transport Protocol using the Audio/Video profile carried over UDP.
  - `VAT` - LBL’s Visual Audio Tool packet format carried over UDP.
  - `UDP` - User Datagram Protocol

  If an application uses a single combined propriety media format and transport protocol over UDP, then simply specifying the transport protocol as UDP and using the format field to distinguish the combined protocol is recommended. If a transport protocol is used over UDP to carry several distinct media types that need to be distinguished by a session directory, then specifying the transport protocol and media format separately is necessary. VAT and RTP are examples of transport protocols that carry multiple payload formats that must be distinguished by the session directory for it to know how to start appropriate tools, relays, mixers or recorders.

  The main reason to specify the transport protocol in addition to the media format is that the same standard media formats may be carried over different transport protocols even when the network protocol is the same - for example vat PCM audio and RTP PCM audio. In addition, relays and monitoring tools that are transport protocol specific but format independent are possible.

  For RTP media streams operating under the RTP Audio/Video Profile [3], the protocol field is ‘‘`RTP/AVP`’’. Should other RTP profiles be defined in the future, their profiles will be specified in the same way. For example, the protocol field ‘‘`RTP/XYZ`’’ would specify RTP operating under a profile whose short name is ‘‘XYZ’’.

- The fourth and subsequent sub-fields are media formats. For audio and video, these will normally be a media payload type as defined in the RTP Audio/Video Profile.

  When a list of payload formats is given, this implies that all of these formats may be used in the session, but the first of these formats is the default format for the session.

  For media whose transport protocol is not RTP or UDP the format field is protocol specific. Such formats should be defined in an additional specification document.

  For media whose transport protocol is RTP, SDP can be used to provide a dynamic binding of media encoding to RTP payload type. The payload names in the RTP AV Profile do not specify unique audio encodings (in terms of clock rate and number of audio channels), and so they are not used directly in SDP format fields. Instead, the payload type number should be used to specify the format for static payload types and the payload type number along with additional encoding information should be used for dynamically allocated payload types.

An example of a static payload type is u-law PCM coded single channel audio sampled at 8KHz. This is completely defined in the RTP Audio/Video profile as payload type 0, so the media field for such a stream sent to UDP port 3456 is:

```
m=video 3456 RTP/AVP 0
```

An example of a dynamic payload type is 16 bit linear encoded stereo audio sampled at 16KHz. If we wish to use dynamic RTP/AVP payload type 98 for such a stream, additional information is required to decode it:

```
m=video 3456 RTP/AVP 98
a=rtpmap:98 L16/16000/2
```

The general form of an `rtpmap` attribute is:

`a=rtpmap:`*<payload type>* *<encoding name>*`/`*<clock rate>*`[/`*<encoding parameters>*`]`

For audio streams, *<encoding parameters>* optionally specifies number of audio channels. For video streams, no encoding parameters are currently specified.

Up to one `rtpmap` attribute can be define for each media format specified. Thus we might have:

```
m=audio 12345 RTP/AVP 96 97 98
a=rtpmap:96 L8/8000
a=rtpmap:97 L16/8000
a=rtpmap:98 L16/11025/2
```

Experimental encoding formats can also be specified in this way. RTP formats that are not registered with IANA as standard format names must be preceded by "`X-`". Thus a new experimental redundant audio stream called GSMLPC using dynamic payload type 99 could be specified as:

```
m=video 3456 RTP/AVP 99
a=rtpmap:99 X-GSMLPC/8000
```

Such an experimental encoding requires that any site wishing to receive the media stream has relevant configured state in its session directory to know which tools are appropriate.

Note that RTP audio formats typically do not include information about the number of samples per packet. If a non-default (as defined in the RTP Audio/Video Profile) packetisation is required, the"`ptime`" attribute is used as given below.

For more details on RTP audio and video formats, see [3].

- Predefined formats for UDP protocol non-RTP media are as below.

  Whiteboard Formats:

  `WB`: LBL Whiteboard (transport: `UDP`)

  Text Formats:

  `NT`: UCL Network Text Editor (transport: `UDP`)

Suggested Attributes

The following attributes are suggested. Since application writers may add new attributes as they are required, this list is not exhaustive.

`a=cat:`*<category>*

This attribute gives the dot-separated hierarchical category of the session. This is to enable a receiver to filter unwanted sessions by category. It would probably have been a compulsory separate field, except for its experimental nature at this time. It is a session attribute.

`a=keywds:`*<keywords>*

Like the cat attribute, this is to assist identifying wanted sessions at the receiver. It is a session attribute.

`a=ptime:`*<packet time>*

This gives the length of time in milliseconds represented by the media in a packet. This is probably only meaningful for audio data. It should not be necessary to know `ptime` to decode RTP or vat audio, and it is intended as a recommendation for the encoding/packetisation of audio. It is a media attribute.

`a=recvonly`

This specifies that the tools should be started in receive-only mode where applicable. It can be either a session or media attribute.

`a=sendrecv`

This specifies that the tools should be started in send and receive mode. This is necessary for interactive conferences with tools such as *wb* which defaults to receive only mode. It can be either a session or media attribute.

`a=sendonly`

This specifies that the tools should be started in send-only mode. Typically this may be used where a different unicast address is to be used for a traffic destination than for a traffic source. It can be either a session or media attribute, but would normally only be used as a media attribute.

`a=orient:`*<whiteboard orientation>*

Normally this is only used in a whiteboard media specification. It specifies the orientation of a the whiteboard on the screen. It is a media attribute. Permitted values are '`portrait`', '`landscape`' and '`seascape`' (upside down landscape).

`a=type:`*<conference type>*

This specifies the type of the conference. Suggested values are '`broadcast`', '`meeting`', and '`moderated`'. 'recvonly' should be the default for '`type:broadcast`' sessions, '`type:meeting`' should imply 'sendrecv' and '`type:moderated`' should imply the use of a floor control tool and that the media tools are started so as to ''mute'' new sites joining the conference.

Specifying the attribute `type:test` is also suggested as a hint that, unless explicitly requested otherwise, receivers can safely avoid displaying this session description to users.

`type` is a session attribute.

`a=charset:`*<character set>*

This specifies the character set to be used to display the session name and information data. By default, an ISO 8859-1 character set is used. If an ISO 8859-1 character set is not suitable, the use of unicode (ISO 10646) [6],[7], as specified in RFC1641 [8] is suggested. In particular, the UTF-7 (RFC1642) [9] encoding is suggested with the following SDP attribute:

```
a=charset:unicode-1-1-utf-7
```

This is a session attribute; if this attribute is present, it must be before the first media field.

`a=framerate:<frame rate>`

This gives the maximum video frame rate in frames/sec. It is intended as a recommendation for the encoding of video data. Decimal representations of fractional values using the notation "<integer>.<fraction>" are allowed. It is a media attribute, and is only defined for video media.

`a=quality:<quality>`

This gives a suggestion for the quality of the encoding as an integer value.

The intention of the quality attribute for video is to specify a non-default trade-off between frame-rate and still-image quality. For video, the value in the range 0 to 10, with the following suggested meaning:

10 - the best still-image quality the compression scheme can give.

5 - the default behaviour given no quality suggestion.

0 - the worst still-image quality the codec designer thinks is still usable.

6.1. Communicating Conference Control Policy

There is some debate over the way conference control policy should be communicated. In general, the authors believe that an implicit declarative style of specifying conference control is desirable where possible.

A simple declarative style uses a single conference attribute field before the first media field, possibly supplemented by flags such as '`recvonly`' for some of the media tools. This conference attribute conveys the conference control policy. An example might be:

```
a=type:moderated
```

In some cases, however, it is possible that this may be insufficient to communicate the details of an unusual conference control policy. If this is the case, then a conference attribute specifying external control might be set, and then one or more "media" fields might be used to specify the conference control tools and configuration data for those tools. A fictional example might be:

```
...
a=type:external-control
m=audio 12345 RTP/AVP 0
m=video 12347 RTP/AVP 31
m=whiteboard 12349 UDP WB
m=control 12341 UDP SCCP
a=mode:chaired
a=chair:128.16.64.2
a=video:follows-audio
a=audio:on-demand
a=audio:chair-mutes-mike
a=whiteboard:chaired
```

In this fictional example (i.e., this is not implemented anywhere), a general conference attribute is specified stating that conference control will be provided by an external tool, and specific attributes are given to specify the control policy that tool should use.

In this document, only the former style of conference control declaration is specified, though we recognise that some variant on the latter may also be used eventually.

Appendix A: SDP Grammar

```
announcement ::=        proto-version
                        origin-field
                        session-name-field
                        information-field
                        uri-field
                        email-fields
                        phone-fields
                        connection-field
                        bandwidth-fields
                        time-fields
                        key-field
                        attribute-fields
                        media-descriptions

proto-version ::=       "v=" (DIGIT)+
                        ;this draft describes version 0

origin-field ::=        "o=" username space
                        sess-id space sess-version space
                        nettype space addrtype space
                        addr newline

session-name-field ::=  "s=" text

information-field ::=   ["i=" text newline]

uri-field ::=           ["u=" uri newline]

email-fields ::=        ("e=" email-address newline)*

phone-fields ::=        ("p=" phone-number newline)*

connection-field ::=    ["c=" nettype space addrtype space
                        connection-address newline]
                        ;a connection field must be present
                        ;in every media description or at the
                        ;session-level

bandwidth-fields ::=    ("b=" bwtype ":" bandwidth newline)*

time-fields ::=         ( "t=" start-time space stop-time
                          (newline repeat-fields)* newline)+
                        [zone-adjustments newline]

repeat-fields ::=       "r=" repeat-interval space typed-time
                        (space typed-time)+

zone-adjustments ::=    time space [``-''] typed-time
                        (space time space [``-''] typed-time)*
```

```
key-field ::=           ["k=" key-type newline]

key-type ::=            "prompt" |
                        "clear:" key-data |
                        "base64:" key-data |
                        "uri:" uri

key-data ::=            printable-ascii

attribute-fields ::=    ("a=" attribute newline)*

media-descriptions ::=  ( media-field
                          information-field
                          connection-field
                          bandwidth-fields
                          key-field
                          attribute-fields )*

media-field ::=         "m=" media space port ["/" integer]
                         space proto (space fmt)+ newline

media ::=               (alpha-numeric)+
                        ;typically "audio", "video", "whiteboard"
                        ;or "text"

fmt ::=                 (alpha-numeric)+
                        ;typically an RTP payload type for audio
                        ;and video media

proto ::=               (alpha-numeric)+
                        ;typically "RTP/AVP", "VAT", or "UDP" for IP4

port ::=                (DIGIT)+
                        ;should in the range "1024" to "65535" inclusive
                        ;for UDP based media ;random allocation should
                        ;only assign above UDP port "5000".

attribute ::=           att-field ":" att-value | att-field

att-field ::=           (ALPHA)+

att-value ::=           (att-char)+

att-char ::=            alpha-numeric | "-"
                        ;is this too tight a restriction

sess-id ::=             (DIGIT)+
                        ;should be unique for this originating username/host

sess-version ::=        (DIGIT)+
                        ;0 is a new session
```

```
connection-address ::=  multicast-conf-address | multicast-scoped-address
                        | unicast-address

multicast-conf-address ::=
                        "224.2." decimal_uchar "." decimal_uchar "/" ttl
                        [ "/" integer ]
                        ;multicast addresses may be in a larger range
                        ;but only these should be assigned by an sdp tool

multicast-scoped-address ::=
                        "239."  decimal_uchar "." decimal_uchar "."
                        decimal_uchar "/" ttl [ "/" integer ]

ttl ::=                 decimal_uchar

start-time ::=          time | "0"

stop-time ::=           time | "0"

time ::=                POS-DIGIT 9*DIGIT
                        ;sufficient for 2 more centuries

repeat-interval ::=     typed-time | interval-time

typed-time ::=          (DIGIT)+ [fixed-len-time-unit]

interval-time ::=       (DIGIT)+ variable-len-time-unit

fixed-len-time-unit ::= ‘‘d’’ | ‘‘h’’ | ‘‘m’’ | ‘‘s’’

variable-len-time-unit ::= ‘‘Y’’ | ‘‘M’’

bwtype ::=              (alpha-numeric)+

bandwidth ::=           (DIGIT)+

username ::=            alpha-numeric
                        ;perhaps this is too restrictive...

email-address ::=       email | email "(" text ")" | text "<" email ">"

email ::=               ;defined in RFC822

uri::=                  ;defined in RFC1630

phone-number ::=        phone | phone "(" text ")" |
                        text "<" phone ">"
```

```
phone ::=                "+" POS-DIGIT (space | "-" | DIGIT)+
                         ;there must be a space or hyphen between the
                         ;international code and the rest of the number.

nettype ::=              "IN"
                         ;list to be extended

addrtype ::=             "IP4" | "IP6"
                         ;list to be extended

addr ::=                 unicast-address

unicast-address ::=      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

text ::=                 (printable-iso8859-1)+ | (unicode-1-1-utf-7)+
                         ;unicode requires a "a=charset:unicode-1-1-utf-7"
                         ;attribute to be used

printable-iso8859-1 ::= ;8 bit ascii character
                         ;decimal 9 (TAB), 32-126 and 161-255

unicode-1-1-utf-7 ::=    unicode-safe
                         ;defined in RFC 1642

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

integer ::= POS-DIGIT (DIGIT)*

alpha-numeric ::=        ALPHA | DIGIT

printable-ascii ::=      unicode-safe | "~" | "\"

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

space ::=               ;ascii code 32
tab ::=                 ;ascii code 9
newline ::=             ;ascii code 10
```

Appendix C: Authors' Addresses

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

Van Jacobson
MS 46a-1121
Lawrence Berkeley Laboratory
Berkeley, CA 94720
United States
electronic mail: van@ee.lbl.gov

Acknowledgments

Many people in the IETF MMUSIC working group have made comments and suggestions contributing to this document. In particular, we would like to thank Eve Schooler, Steve Casner, Bill Fenner, Allison Mankin and Ross Finlayson.

References

[1] D. Mills, "Network Time Protocol version 2 specification and implementation", RFC1119, 1st Sept 1989.

[2] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889

[3] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 1890

[4] M. Handley, "SAP - Session Announcement Protocol", INTERNET-DRAFT, November 25th 1996.

[5] V. Jacobson, S. McCanne, "vat - X11-based audio teleconferencing tool" vat manual page, Lawrence Berkeley Laboratory, 1994.

[6] "The Unicode Standard, Version 1.1": Version 1.0, Volume 1 (ISBN 0-201-56788-1), Version 1.0, Volume 2 (ISBN 0-201-60845-6), and "Unicode Technical Report #4, The Unicode Standard, Version 1.1" (available from The Unicode Consortium, and soon to be published by Addison-Wesley).

[7] ISO/IEC 10646-1:1993(E) Information Technology--Universal Multiple-octet Coded Character Set (UCS).

[8] D. Goldsmith, M. Davis, "Using Unicode with MIME", RFC1641, July 1994

[9] D. Goldsmith, M. Davis, "UTF-7 - A Mail-Safe Transformation Format of Unicode", RFC1642, July 1994