Internet Engineering Task Force **MMUSIC WG**
INTERNET-DRAFT **Mark Handley/Van Jacobson**
draft-ietf-mmusic-sdp-01.ps **UCL/LBL**
22nd Nov 1995
Expires: 22nd May 1995

SDP: Session Description Protocol

Status of this Memo

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

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

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

Distribution of this document is unlimited.

Abstract

The *sd* session directory tool has been in use for some time on the Mbone for announcing multicast sessions. This document describes an enhanced version of the sd protocol (SDP v2), and explains the extensions to the protocol that have become desirable.

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

The LBL session directory tool (*sd*) has been in use on the Mbone for some time to advertise multimedia conferences and communicate the conference addresses and conference tool specific information necessary for participation. This document defines an extended version of the session directory protocol and some extensions to the protocol that have become desirable. In the body of the paper, we describe a proposed Session Description Protocol (SDP v2). In Appendix A, we describe how this differs from the protocol currently used by sd (SDP v1). This draft does not describe multicast address allocation or the distribution of SDP messages in detail - these are left to accompanying drafts.

2. Background

The Multicast Backbone (Mbone) is an experimental overlay network on the internet that permits efficient many to many communication. For the past few years it has been used extensively for multimedia conferencing. Such multimedia conferences usually have the property that tight coordination of conference membership is not necessary; in order to receive a conference, a user at an Mbone site has to know only the correct multicast group address for the conference and the UDP ports the conferencing applications will use to receive the conference data streams.

In order to assist the advertisement of conference sessions and to communicate the relevant conference setup information to prospective participants, the *session directory* (sd) tool was written. Sd has now been in wide scale use for close to 2 years, during which time Mbone usage has greatly increased and diversified. The Mbone has now reached the stage where assistance with coordination of resource usage is required, and where compatible session announcement tools are starting to emerge.

This document is an attempt to prevent diversification of the sd protocol as tool writers each add their own modifications. It is also an attempt to provide guidelines to the writers of such announcement tools in order to protect the Mbone from misuse and to preserve the inherent scalability of the original sd program whilst enhancing its functionality.

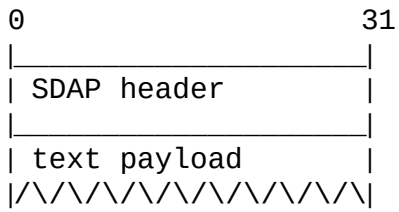
In defining SDP v2, we also aim to enhance to generality of SDP so that it can be used for a wider range of network environments and applications.

3. The Use of SDP (background)

3.1. Multicast Announcement

SDP is a session description protocol for multimedia sessions. It is normally used by an SDP client which 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 sd clients will need to be able to listen for such announcements on multiple addresses. The Session Directory Announcement Protocol is described in more detail in a companion draft.

Sd packets are UDP packets of the following format:



The first four bytes are Session Directory Announcement Protocol (SDAP) header.

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 SDAP, only one session annoucement is permitted in a single packet.

3.2. SDP announcement by email and WWW

It should be noted that announcements of multicast sessions made only via email or the World Wide Web (WWW) do not have property that the receiver of a session announcement can receive the session, nor do they provide Mbone booking feedback or allow scalable dynamic multicast address allocation, and so should normally be used to supplement periodic multicast announcements.

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

4. Requirements

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 a duration of time. Media streams can be many-to-many. The times during which the session is active need not be continuous.

Multicast based sessions on the internet differ 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 - as a means to communicate the existence of a session, and as 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 the information SDP must convey includes:

- Name and purpose of session
- Time(s) the session is active
- The media comprising the session
- Information to receive those media

As resources (such as bandwidth) necessary to participate in a session may be limited, some additional information is also desirable:

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

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.

4.1. Media Information

The information that must be conveyed is:

- 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)

In an IP multicast session, the following must also be conveyed:

- Multicast address for media
- Transport Port for media

In an IP unicast session, the following must be 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 networks will have their own specific requirements - SDP must be extensible for these.

4.2. Timing Information

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

SDP must be able to 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"
- These times must be globally consistent, irrespective of local time zone or daylight saving time

4.3. Private Sessions

It should be possible to create both public sessions and private sessions. However, private sessions on the existing Mbone infrastructure rapidly use up the available bandwidth. It should be possible to create private sessions along with contact information if those sessions become a problem.

If a session announcement is private (encrypted) it should be 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.

4.4. Further Information

SDP should convey enough information to decide whether a session is the session a user wishes to participate in. It should also convey where to go to find more information about the session. This extra information should be in the form of Universal Resources Identifiers (URIs).

4.5. Categorisation

When many session descriptions are being conveyed by SDAP or any other advertisement mechanism, it is important to be able to filter session announcements that are of interest from those that are not. SDP should support a categorisation mechanism for sessions that can be automated.

4.6. 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.

5. 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 process session descriptions. However, since the total bandwidth allocated to all SDAP 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 likely 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 takes the form of a number of lines of text of the form
<*type*>=<*value*>
<*type*> is always exactly one character and case is 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 free format string.

Each announcement consists of a session description section followed by zero or more 'media' description sections. The session description starts with an '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 SDAP, only one session description is allowed in each packet. When SDP is conveyed by other means, many SDP session descriptions may be carried 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=* (URL of description)
- e=* (email address)
- p=* (phone number)
- c= (connection information)
- b=* (bandwidth information)
- t=* (zero or more times)
- k=* (encryption key)
- a=* (zero or more session attribute lines)

Media description
- m= (media name and transport address)
- i=* (media title)
- c=* (connection information)
- 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 sd 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. In the example below, each media behaves as if it were given a ‘`recvonly`’ attribute.

An example SDP v2 description is:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=Sd Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.01.ps
e=M.Handley@cs.ucl.ac.uk (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 3456 VAT PCMU
m=video 2232 RTP H261
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.

SDP version 1 is not compatible with SDP version 2 described below, although it is similar. The differences between SDP v1 and SDP v2 are described in Appendix A.

Protocol Version

`v=0`

The ‘‘v’’ field gives the version of the Session Description Protocol. As SDP v1 had no version number, we begin numbering with SDP v2 as protocol version 0. 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 their 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. <*session id*> is a numeric string such that the triple of <*username*>, <*session id*> and <*address*> form a globally unique identifier for the session. Its method of allocation is up to the creating tool, but it has been suggested that a Network Time Protocol (NTP, [1]) timestamp be used to ensure uniqueness. <*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 has been suggested (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 that the session was created from. 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).

Information

`i=`<*session description*>

The ‘‘i’’ field is information about the session. There must be no more than one ‘‘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
  ```

- 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=M.Handley@cs.ucl.ac.uk (Mark Handley)
  ```

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

  ```
  e=Mark Handley <M.Handley@cs.ucl.ac.uk>
  ```

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

  Conferences using a IP multicast connection address must also have a TTL (time to live) 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., 47 for all sites within an organization).

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

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

  The RSVP WG of the IETF has defined that hierarchical encoding schemes should be transmitted in multiple multicast groups to allow multicast pruning to keep unwanted traffic from sites only requiring some levels of the hierarchy. For applications which require 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 for either ttl or number of addresses to be used for IP unicast addresses.

  A session announcement must contain at least one ‘‘c’’ field. It may contain one additional ‘‘c’’ field per media field (see below), in which case the per-media values override the conference-wide 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* 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 in the ttl, 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*: 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 applications ''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.

It is unclear who (if anyone) should be the registry for bandwidth modifiers.

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 addition period of time that the session will be active for. If the session is active at regular times, an ''r'' field should be used in addition to 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, [1]) time values in seconds. To convert these values to UNIX time, subtract decimal 2208988800. If these values are both set to zero, then the conference is not bounded. User interfaces should prohibit or strongly discourage the creation of unbounded conferences as they give no information about when the session is actually going to be active. 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'' field would be:

```
r=604800 3600 0 90000
```

To make announcements more compact, times may also be given in units of days, hours or minutes. To allow yearly or monthly announcements (same day each year or month), units of years and months are also allowed. 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:

`Y` - years (same day of same month each repeated year)
`M` - months (same day of the month each repeated month)
`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 have been written:

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

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

Should it be necessary to schedule a repeated session which spans a change from daylight 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, because different countries change to or from daylight time on different dates, and because 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
```

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=`<*encryption key*>

- In countries where encrypted sessions are not prohibited by law, the session description protocol may be use to convey encryption keys.
- A key field is permitted before the first media entry, or for each media entry as required.
- The format of keys and their usage is outside the scope of this document, but see [4]

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’’.

- 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 sd 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*>

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’’ and ‘‘text’’, 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 RTPv2 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 send 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*>

  In such a case, the ports used depend on the transport protocol. For RTPv2, 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 H261
  ```

  would specify that ports 3456 and 3457 form one RTP/RTCP pair and 3458 and 3459 form the second RTP/RTCP pair.

  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` - the IETF's Realtime Transport Protocol carried over UDP.
- `VAT` - LBL's Visual Audio Tool packet format carried over UDP.
- `UDP` - User Datagram Protocol

If an application uses a propriety media format and transport protocol over UDP, then simply specifying the transport protocol as UDP is recommended.

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.

- The fourth sub-field is the media format. For audio and video, this will normally be a media format string as defined in the RTP Audio/Video Profile.

Predefined formats are as below. For more details on audio and video formats, see [3].

- **Audio Formats:**

  `PCMU`: 8-bit mu-law encoded 8kHz PCM
  `PCMA`: 8-bit A-law encoded 8kHz PCM
  `IDVI`: Intel DVI ADPCM
  `GSM`: GSM (Group Speciale Mobile)
  `LPC`: An experimental Linear Predictive Coder written by Ron Frederick
  `1016`: CELP encoding as specified in FED-STD 1016
  `G721`: ITU recommendation G721
  `G723`: ITU recommendation G723
  `L8`: 8 bit linear audio
  `L16`: 16 bit linear audio
  `MPA`: MPEG audio
  `VDVI`: A variable rate version of IDVI
  `TSP0`: TrueSpeech proprietary encoding
  `VSC`: Vocaltec Software Compression proprietary encoding.

- **Video Formats:**

  `JPEG`: Motion JPEG
  `MPV`: MPEG encoding
  `MP2T`: MPEG II transport stream
  `CelB`: Sun Cell-B encoding
  `H261`: CCITT/ITU-T recommendation H.261
  `nv`: Xerox Parc Network Video
  `CPV`: Compressed Packet Video (proprietary encoding)

HDCC: HDCC proprietary encoding from Silicon Graphics

CUSM: CU-SeeMe video encoding

PicW: PictureWindow encoding from BBN

RGB: 8 bit encoding of RGB values

- **Whiteboard Formats:**

  WB: LBL Whiteboard (transport: UDP)

- **Text Formats:**

  NT: UCL Network Text Editor (transport: UDP)

  MMBL: mumble text chat tool (transport: UDP)

- Note that audio formats do not include packetisation information. If a non-default (as defined in the RTP Audio/Video Profile) packetisation is required, the ''`ptime`'' attribute is used as given below.

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 - 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 that 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 only used in a whiteboard media specification, this 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. It 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.

5.1. 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 VAT PCMU
m=video 12347 RTP H261
m=whiteboard 12349 UDP WB
m=control 12341 UDP CCCP
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

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

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

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

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

key-field ::=           ["k=" (printable-ascii)+ newline]
```

```
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 media type for audio
                        ;and video media

proto ::=               (alpha-numeric)+
                        ;typically "RTP", "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 ::=            ;not defined here

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 B: Summary of differences between SDPv1 and SDPv2

For this purpose SDPv1 is defined as the protocol in use by version 1.14 of the sd session description tool. SDPv2 is the proposed protocol described in the document.

SDPv1 allows the use of only the following SDP fields:

- `s=` **session name** - no change
- `i=` **session information** - SDP v2 allows ‘‘i’’ fields to be used in the media descriptions as media labels.
- `o=` **originator** - in SDP v1, originator is of the form *username@hostname*, and the sd packet headers contain a binary representation of the IP 4 address of the originating host. This essentially contains duplicate information and cannot cope with non-IP4 addresses. Thus in SDP v2 we add an address type field and put the address of the originating machine in the originator field. In SDP v2 we also add two more sub-fields - a session id and a session version - to the origin field. This means that all the information to identify a session and whether that session has changed is in one field. To make this field easier to find, we move to to the beginning of the announcement after the new protocol version field.
- `c=` **conference data** - only one conference data field is allowed before the first media field in SDPv1. In SDP v1 the conference start and stop times are the third and fourth fields of the conference data field. In SDP v2 they have moved to the `t=` time field. To convert from SDPv1 time-stamps to UNIX time, subtract decimal 2085978496. To convert from SDPv2 time-stamps to UNIX time, subtract decimal 2208988800 (SDPv1 uses NTP time-stamps incorrectly).

  In SDPv1 no network type or address type subfields are present, and the ttl is a separate subfield following the multicast address. Multiple multicast addresses are not allowed in SDPv1.
- `m=` **media** - In SDP v1, the third subfield is the RTP v1 ID. This is now obsolete. In SDP v1, the media format allowed a default format if none was specific, but non-default values were specified using the `fmt` attribute. In SDP v2, allow default attributes and not allowed, and the fourth subfield in a media field gives the media format. In SDPv1, there is no way to distinguish between the same media format carried by different transport protocols. In SDP v2 the third subfield in a media field gives the transport protocol.

  In SDPv1 multiple ports are not allowed for hierarchical encodings on a unicast address.
- `a=` **attributes** - attributes are allowed only after the first media field in SDPv1.

SDPv2 additionally defines the following fields which are defined above:

- `v=` **Protocol Version**
- `b=` **Bandwidth**
- `e=` **Email Address**
- `p=` **Phone Number**
- `u=` **URL**
- `t=` **Time**
- `r=` **Repeat Time**

z= **Time Zone info**

k= **Encryption Key**

Appendix C: Authors' Addresses

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

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

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’’, INTERNET-DRAFT, draft-ietf-avt-rtp-07.txt, 21st March 1995.

[3] H. Schulzrinne, ‘‘RTP Profile for Audio and Video Conferences with Minimal Control’’, INTERNET-DRAFT, draft-ietf-avt-profile-04.txt, 21st March 1995.

[4] M. Handley (editor), ‘‘The Use of Plain Text Keys for Multimedia Conferences’’, Research Note RN-95-19, Department of Computer Science, University College London, Feb 1995. http://www.cs.ucl.ac.uk/rns/RN9519.ps

[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