rtcweb D. R. Worley
Internet-Draft Ariadne
Intended status: Standards Track February 25, 2013
Expires: August 29, 2013

Kumquat: A Generic Bundle Mechanism for the Session Description Protocol (SDP)

draft-worley-sdp-bundle-04

Abstract

This document defines a generic bundle mechanism for the Session Description Protocol (SDP) by which the media described by a number of media descriptions ("m= lines") are multiplexed and transmitted over a single transport association. The transport association is described by an additional media description, allowing SDP attributes to be applied to the aggregate, independently of attributes applied to the constituents. In offer/answer usage, the bundle mechanism is backward compatible with SDP processors that do not understand the mechanism. The mechanism is designed to be compatible with the limitations of the existing Internet infrastructure.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The central idea of bundling is to multiplex the media that would be several RTP sessions into one RTP session, with particular emphasis on allowing one transport association to carry media that are presented to the higher, application layer, as multiple RTP sessions.

At the interface between the SDP-configured layer and the lower, transport layer, the media are organized into a single RTP session. The transport-related properties of the RTP session (e.g., transport 5-tuple, encryption, ICE) are described by the transport-related attributes of a single media description.

At the interface between the SDP-configured layer and the higher, application layer, the media are organized into several RTP sessions. The application-related properties of the RTP session (e.g., media type and label) are described by the application-related attributes of separate media descriptions.

(There are some attributes (e.g., bandwidth limitation) that can apply separately to both the bundled RTP session and the constituent RTP sessions.)

However, we do not include the payload type numbers as information available to the application; only the encoding name and its parameters are accessible to the application. This gives the bundle mechanism freedom to place constraints on the use of payload types.

The bundle is signaled in the session description by a "group" attribute with semantics "KUMQUAT". The first media description

listed in the group is the "bundle" media description (MD), whose transport information describes the transport association via which the RTP packets will be sent. The remaining (zero or more) media descriptions listed in the group are the "constituent" MDs. RTP packets received from the applications for these MDs are encapsulated and sent on the transport association for the bundle MD. RTP packets received from the transport association for the bundle MD are deencapsulated and sent to the applications for the constituent MDs.

A new payload type (codec) named "kumquat" is defined to be used for this encapsulation. Section 5.4.1

In offer/answer usage, we must arrange that the bundle mechanism is backward compatible with entities that do not understand the bundle mechanism. This requirement drives many features of this solution. Section 6.1

In addition, many devices in current usage (especially SBCs) apply more restrictions on the usage of SDP than one would expect from abstract consideration of their roles in the network. Some features of this solution are constructed to avoid these restrictions. Section 6.2

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The important RFCs in this area use inconsistent terminology. Here, we use:

media Media is (1) media content, considered in an abstract way, that is, without consideration of its particular encoding or the framing information around it, and (2) the particular bits and octets used to encode and transmit the abstract media content.

media stream (Taken from [RFC3550].) A media stream is a set of RTP packets that are generated by and interpreted by one codec. The RTP packets of a media stream are identified by a unique SSRC.

capture (Taken from CLUE's work.) A capture is a set of media streams that originate from one (physical or virtual) media source and should be composed to provide rendering of that source. For example, media streams from one origin including layered encodings, forward error correction streams, recovery streams, and simulcasted media streams of varying bit rates compose one capture.

transport association (Taken from [I-D.alvestrand-mmusic-msid].) A transport association is a single data path between two hosts, such as a TCP connection, or a pair of UDP ports that send packets to each other. A transport association is identified by the identity of the protocol being used, the relevant host addresses, and the relevant port numbers. In the case of unicast communications, these form a "5-tuple", namely, the protocol, the host addresses of the two hosts, and the port numbers used on the two hosts. In the case of multicast sessions, these form a "3-tuple", namely, the protocol, the multicast address, and the port number. In SDP, a transport association is specified by the address and port of a media description (and possibly the same information from the matching offer/answer SDP). If a media description specifies multiple addresses or ports, each address or port specifies one transport association.

transport flow (Taken from [I-D.ietf-avtcore-multi-media-rtp-session].) (This is called an "RTP session" by [RFC3264].) A transport flow is the data that flows across a transport association.

media description (Taken from [RFC4566].) A media description is one group of lines in a session description demarcated by an m= line. By synecdoche, a media description is often referred to as "an m= line".

transport association group A transport association group is the set of transport associations denoted by one media description. Usually the m= line specifies only one port and the c= line specifies only one address, and so the media description's transport association group contains only one transport association.

transport flow group A transport flow group is the set of transport flows of the transport associations of a transport association group.

session description (Taken from [RFC4566] section 2.) A session description is an SDP instance.

multimedia session (Taken from [RFC4566] section 2.) A multimedia session is the totality of the media that is transmitted/received as described by a particular session description.

RTP session (Taken from [RFC3550].) An RTP session is a group of media streams which must not have duplicated SSRC values because the endpoints share RTCP reporting information. Note that an RTP session may encompass more than one multimedia session. RTP sessions are not fully described by session descriptions.

3. Desiderata

This section lists desiderata for the bundle mechanism in SDP. (I use the term "desiderata" -- "things that are desired" -- rather than "requirements", because we may discover that we can't optimally satisfy all of these criteria at the same time.) The first section lists desiderata that are arise from considering the ways applications may wish to bundling. The second section lists desiderata that arise from compatibility with existing Internet infrastructure.

3.1. Feature Desiderata

These desiderata describe features that we would like the bundling mechanism to provide.

DES F1 For each bundle, there is a group of media descriptions which describe the application-level RTP sessions. This specification must allow the same granularity of description as when the media flows were not multiplexed. This description includes identifiers which connect the media flows with the application and with each other.

This requirement is taken from [I-D.jennings-mmusic-media-req].

DES F2 For each bundle, there is a media description that describes the transport-level RTP session.

DES F1 and DES F2 do not specify whether the transport-level media description may or may not also be one of the application-level media descriptions.

DES F3 There must be a uniform way to deal with new SDP parameters, so that newly defined SDP parameters do not require a specific updating of the bundling procedures.

This desideratum is taken from slides-interim-2013-rtcweb-1-10.pdf.

DES F4 Multiple separate bundles within one SDP must be supported.

DES F5 Bundles may contain other bundles as constituents.

Of course, no bundle may directly or indirectly contain itself. (I don't expect any current implementation to implement bundles within bundles, but we should design the mechanism to allow this, as some day we will likely need it.)

DES F6 A bundle may contain zero constituents.

A bundle with no constituents serves no purpose for the transport of media, but we are likely to someday need to describe such a bundle. (Compare that an SDP m= line is syntactically constrained to specify at least one payload type. When SDP was used only to specify multicast sessions, this constraint was common sense. But once SDP offer/answer was invented, when a media description was rejected, the natural representation would be an m= line with a zero port and no payload types. But a payload type was syntactically required, so we now have to provide at least one token payload type in rejected m= lines.)

DES F7 If an answerer that does understand the bundle mechanism processes an offer that contains a bundle, it must be able to (1) accept the bundle and selectively accept or reject each constituent RTP session within it, (2) reject the bundle as a whole, or (3) reject the bundling and selectively accept or reject each constituent RTP session as separate RTP sessions.

Presumably answer (3) resembles that which would be produced by an answerer that does not understand the bundle mechanism. It is a lower priority that the answerer can distinguish between accepting the bundle while rejecting all of its constituents, and rejecting the bundle as a whole. But those two conditions differ conceptually regarding whether any "framing" actions of the bundle are performed.

DES F8 There must be a reliable way to demultiplex incoming RTP into the separate application-level RTP sessions. Similarly, there must be a reliable way to demultiplex the associated RTCP information.

The RTCP information for each media stream is tagged with the SSRC about which it reports, and the SSRC is used to correlate the RTCP reports with the RTP sessions containing media with the same SSRC. So regarding RTCP, this desideratum appears to be straightforward to satisfy.

DES F9 The specification must specify any needed additional procedures for handling SSRC collisions between media sources within different application-level RTP sessions, as those can now collide.

In the terminology of [RFC3550], the constituent media descriptions are now part of one RTP session.

DES F10 When an offer is constructed, the offerer must not need to preallocate TURN relays for constituent media descriptions. When both endpoints support bundling, the mechanism must not require the offerer to allocate TURN relays for constituent media descriptions.

This desideratum was suggested by Andrew Hutton.

DES F11 It must be possible to add and remove one way video flows within the bundle without requiring an additional offer/answer cycle.

Presumably this can be accomplished as it is now, with a single media description carrying multiple video flows that are distinguished only by their SSRCs. This desideratum is taken from slides-interim-2013-rtcweb-1-10.pdf.

DES F12 Bundling must not interfere with ICE usage, and in particular, ICE's ability to negotiate both IPv4 and IPv6 addresses simultaneously.

This desideratum was suggested by Andrew Hutton.

3.2. Compatibility Desiderata

These desiderata describe compatibility of the bundling mechanism with with non-supporting endpoints or with existing entities in the Internet infrastructure.

DES C1 In offer/answer usage, an endpoint using the bundle mechanism must interwork correctly with an endpoint that does not understand the bundle mechanism.

DES C2 Interworking must continue when SDP endpoints are replaced with other endpoints during a sequence of offer/answer exchanges (such as happens in 3PCC or call transfers "behind an SBC"), including when a supporting endpoint is replaced by a non-supporting endpoint or vice-versa.

SDP features (e.g., the codec set and ICE) are generally designed so that an offerer always offers every facility it is willing to support in the current situation, regardless of whether it was agreed to by the answerer in a preceding exchange. Thus, if the current answerer is a different endpoint than the previous answerer, the new answerer will negotiate a compatible set of facilities without needing

knowledge of its predecessor's SDP. The offerer will smoothly transition to the new facilities. This property is required to support 3PCC situations (e.g., [RFC3725] and [I-D.worley-service-example]). This desideratum was suggested by Richard Ejzak.

DES C3 Avoid using media types in m= lines other than audio and video unless required for user media, as some SBCs reject SDP that uses other media types.

This desideratum was suggested by Hadriel Kaplan.

DES C4 Any additional m= lines prescribed by the bundle mechanism should be ordered after the constituent m= lines.

Many devices that have only one audio or video channel accept the first m= line with that media type and reject any further ones

non-DES C5 SBCs generally pass through attributes that they do not understand. SBCs generally pass through codec specifications that they do not understand, even if they are configured to transcode certain specific codecs.

This non-desideratum was suggested by Hadriel Kaplan.

DES C6 After offer/answer processing is finished, if the exchanged SDP is examined by a non-supporting SBC, the set of transport associations that it sees being specified for media exchange should be the set that are actually used for media transfer.

This is needed because SBCs monitor the packet traffic on the transport associations and if no media is seen on one of the associations for a significant period of time, the SBC will tear down the call. This desideratum was suggested by Hadriel Kaplan.

DES C7 In a session description, no endpoint of a transport association may be used multiple times.

Such duplication is not defined by [RFC4566]. Some SBCs do not support such duplication (ultimately, because it was not supported by [RFC2327]), and they reject SDP specifying duplicated transport association endpoints. This desideratum was suggested by Cullen Jennings.

DES C8 Offer/answer processing between supporting processors must be completed in one exchange. When interworking between supporting and non-supporting processors, it is less desirable but admissible that a second offer/answer exchange may be needed to complete configuring the multimedia session.

DES C9 If an intermediate entity provides transcoding between codecs, and if the offer/answer does not specify encryptioon of a media stream, the media stream should be able to take advantage of the transcoding facility.

4. Tutorial Examples

This section is non-normative. (This section was suggested by Charles Eckel.)

This is an introduction to SDP bundling via a series of examples of offer/answer processing. Some mandatory SDP lines have been omitted from the examples for brevity. Long SDP lines have been folded by using trailing backslashes. Blank lines have been inserted for clarity.

4.1. One Audio Stream and One Video Stream

4.1.1. Offer without Bundling

Here is a typical, non-bundled SDP example with both audio and video media:

```
        o=- 2890844526 2890844526 IN IP4 host.example.com
        c=IN IP4 10.0.1.1
```

This SDP media description (MD) provides the transport information about the audio and also identifies the role of the audio from the application's point of view. In this case, the fact that it is the first audio m= line suffices to tell the application how to treat it. In more complex cases, label or content attributes might be used to communicate the proper handling to the application.

```
        m=audio 10000 RTP/AVP 0 8 97
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10000 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51000 typ srflx \
            raddr 10.0.1.1 rport 10000
```

This MD provides the transport information about the video and also identifies the role of the video from the application's point of view.

```
        m=video 10002 RTP/AVP 31 32
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10002 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51002 typ srflx \
            raddr 10.0.1.1 rport 10002
```

We call the RTP that is described by each media description (MD) a transport flow (TF). The audio and video are carried in separate TFs, which each have a separate transport association (address/port).

4.1.2. Offer with Bundling

With SDP bundling, we add an additional MD to describe a single "bundle" TF to carry both the audio and video information, and a group attribute to show the association of the bundle MD with the constituent MDs:

```
        o=- 2890844526 2890844526 IN IP4 host.example.com
        c=IN IP4 10.0.1.1
```

Declare which MDs are included in the multiplexed MD: mid:con1 and mid:con2 are the constituent MDs whose TFs (from the application point of view) will be carried by the TF of the first-designed MD, mid:bundle, which is the bundle MD.

```
        a=group:KUMQUAT bundle con1 con2
```

This MD provides the application-level description of the audio TF. As in the previous example, it is the first audio m= line. It includes any attributes which apply to the audio media from the application point of view, including the payload type definitions. When interpreted by a supporting processor, the transport information is ignored. When interpreted by a non-supporting processor, the transport information specifies that the TF exists but is currently "on hold": the association address is null, and the association port is 9, the discard port.

```
        m=audio 9 RTP/AVP 0 8 97
        c=IN IP4 0.0.0.0
        a=mid:con1
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
```

```
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000
```

This MD provides the application-level description of the video TF. As in the previous example, it is the first video m= line. It includes any attributes which apply to the video media from the application point of view. As in the audio MD, the association address is null, and the association port is 9.

```
        m=video 9 RTP/AVP 31 32
        c=IN IP4 0.0.0.0
        a=mid:con2
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000
```

This MD provides the transport information for the bundle TF, including any attributes which apply to the transport. We use RTCP multiplexing [RFC5761], so only one set of ICE candidates (and only one TURN relay) is needed for each MD. The MD is artificially given the media type "audio" (which is ugly, but it avoids rejection by SBCs) and it is placed after all of the constituent MDs so as to not affect their positions as "first audio MD", etc. The MD lists a single payload type for the "kumquat" payload format, which is used to encapsulate the RTP of the constituent TFs.

```
        m=audio 10000 RTP/AVP 127
        a=mid:bundle
        a=rtcp-mux
        a=rtpmap:127 kumquat
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10000 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51000 typ srflx \
            raddr 10.0.1.1 rport 10000
```

If this SDP bundle is accepted, RTP provided by the application for the audio TF will be encapsulated into a kumquat payload and then be sent from port 10000. The encapsulation also contains the ordinal index (i.e., 0) of the audio TF and the payload type of the original audio RTP. RTP provided by the application for the video TF will be encapsulated into a kumquat payload and then be sent from port 10000. The encapsulation also contains the ordinal index (i.e., 1) of the video TF and the payload type of the original video RTP.

RTP that is received on port 10000 is interpreted according to the kumquat payload format: The constituent MD ordinal index is extracted. The encapsulated RTP and its payload type are then interpreted according to the constituent MD.

4.1.3. Answer from an Answerer that Supports Bundling

If the answerer supports SDP bundling, and desires to accept the offered bundle and its constituent MDs, the answerer signals that it accepts the SDP bundling by providing a matching group:KUMQUAT attribute in the answer. As always in offer/answer, the MDs in the answer correspond to the MDs in the offer by ordinal position.

The answerer provides the necessary transport information for the bundle MD. The answerer understands that MDs mid:con1 and mid:con2 are incorporated into MD mid:bundle, and ignores their transport information. It accepts each constituent MD by providing an answer MD for each of them that specifies a null address and port 9 (the discard port).

```
        o=- 2890844526 2890844526 IN IP4 answer.example.com
        c=IN IP4 10.0.2.1

        a=group:KUMQUAT bundle con1 con2

        m=audio 9 RTP/AVP 0 8 97
        c=IN IP4 0.0.0.0
        a=mid:con1
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000

        m=video 9 RTP/AVP 31 32
        c=IN IP4 0.0.0.0
        a=mid:con2
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000

        m=audio 20000 RTP/AVP 127
        a=mid:bundle
        a=rtcp-mux
        a=rtpmap:127 kumquat
        a=candidate:0 1 UDP 2113601791 10.0.2.1 20000 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.35 51090 typ srflx \
            raddr 10.0.2.1 rport 20000
```

4.1.4. Answer from an Answerer that Does Not Support Bundling

SDP bundling allows for backward compatibility in case the answerer does not understand bundling. If the answerer does not understand

bundling, it ignores the group attribute, and effectively sees the offer as:

```
        o=- 2890844526 2890844526 IN IP4 host.example.com
        c=IN IP4 10.0.1.1

        m=audio 9 RTP/AVP 0 8 97
        c=IN IP4 0.0.0.0
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000

        m=video 9 RTP/AVP 31 32
        c=IN IP4 0.0.0.0
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000

        m=audio 10000 RTP/AVP 127
        a=rtcp-mux
        a=rtpmap:127 kumquat
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10000 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51000 typ srflx \
            raddr 10.0.1.1 rport 10000
```

If the answerer wishes to accept the first audio and video streams, it assembles this answer:

```
        o=- 2890844526 2890844526 IN IP4 answer.example.com
        c=IN IP4 10.0.2.1
```

The absence of the group attribute informs the offerer that bundling was rejected.

The audio MD is accepted. Transport information is provided, but it does not include ICE candidates, because the offer did not provide ICE candidates for the first and second MDs.

```
        m=audio 20000 RTP/AVP 0 8 97
        c=IN IP4 10.0.2.1
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000
```

The video MD is accepted. Transport information (using a different

port) is provided.

```
        m=audio 20002 RTP/AVP 31 32
        c=IN IP4 10.0.2.1
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000
```

The bundle MD is rejected by the answerer because the only offered codec was kumquat, and the answerer does not implement it.

```
        m=audio 0 RTP/AVP 127
```

Because the group attribute is not present in the response, the offerer knows that the answerer does not support bundling (or does not want to consider the offered bundle). The offerer knows that the answerer wants to establish one audio TF and one video TF, and formally, that has been done. But the offerer has not set up its transport for separate audio and video TFs and has not signaled its transport information for those TFs to the answerer.

In order to enable media flow, the offerer sends an updated offer containing transport information for the constituent MDs:

```
        o=- 2890844526 2890844527 IN IP4 host.example.com
        c=IN IP4 10.0.1.1
```

No group attribute is included, to ensure that this update only sets transport attributes, and does not trigger bundle-supporting behavior if the answering entity has changed in the meantime.

Provide transport attributes for the audio MD. (We can reuse the ICE candidates (and TURN relay) offered for the bundle MD.)

```
        m=audio 10000 RTP/AVP 0 8 97
        c=IN IP4 10.0.1.1
        a=mid:con1
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10000 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51000 typ srflx \
            raddr 10.0.1.1 rport 10000
```

New ICE candidates (and a separate TURN relay) are needed for the video MD.

```
        m=video 10002 RTP/AVP 31 32
        c=IN IP4 10.0.1.1
        a=mid:con2
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10002 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51002 typ srflx \
            raddr 10.0.1.1 rport 10002
```

The bundle MD must still be listed, but it is disabled.

```
        m=audio 0 RTP/AVP 127
        a=mid:bundle
```

The answerer then provides an answer that contains ICE candidates:

```
        o=- 2890844526 2890844527 IN IP4 answer.example.com
        c=IN IP4 10.0.2.1

        m=audio 20000 RTP/AVP 0 8 97
        c=IN IP4 10.0.2.1
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000
        a=candidate:0 1 UDP 2113601791 10.0.2.1 20000 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.35 51090 typ srflx \
            raddr 10.0.2.1 rport 20000

        m=audio 20002 RTP/AVP 31 32
        c=IN IP4 10.0.2.1
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000
        a=candidate:0 1 UDP 2113601791 10.0.2.1 20002 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.35 51092 typ srflx \
            raddr 10.0.2.1 rport 20002

        m=audio 0 RTP/AVP 127
```

The ICE negotiations proceed, the transport associations are established, and RTP flows.

4.1.5. Fast-Start Offer

The basic procedure requires the offerer to update its offer when it discovers that the answerer does not support SDP bundling. The offerer can avoid this delay by providing transport information for the constituent MDs as well as for the bundle MD. The penalty is that the offerer must preallocate TURN relays for both the constituent MDs as well as the bundle MD.

```
        o=- 2890844526 2890844526 IN IP4 host.example.com
        c=IN IP4 10.0.1.1

        a=group:KUMQUAT bundle con1 con2

        m=audio 10000 RTP/AVP 0 8 97
        c=IN IP4 10.0.1.1
        a=mid:con1
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10000 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51000 typ srflx \
            raddr 10.0.1.1 rport 10000

        m=video 10002 RTP/AVP 31 32
        c=IN IP4 10.0.1.1
        a=mid:con2
        a=rtcp-mux
        a=rtpmap:31 H261/90000
        a=rtpmap:32 MPV/90000
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10002 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51002 typ srflx \
            raddr 10.0.1.1 rport 10002

        m=audio 10004 RTP/AVP 127
        c=IN IP4 10.0.1.1
        a=mid:bundle
        a=rtcp-mux
        a=rtpmap:127 kumquat
        a=candidate:0 1 UDP 2113601791 10.0.1.1 10004 typ host
        a=candidate:1 1 UDP 1694194431 198.51.100.32 51004 typ srflx \
            raddr 10.0.1.1 rport 10004
```

If the answerer understands bundling and accepts the bundle, it rejects the constituent MDs and accepts the bundle MD. If the answerer does not understand bundling, it accepts the constituent MDs and rejects the bundle MD.

4.2. Two Audio Streams and Two Video Streams

In this example, a presentation involves four media roles: the speaker's audio, the floor microphone, the video of the speaker, and the video of the speaker's slides. We use separate MDs for each media stream because each TF has a different role; the application will handle each of them in distinctly different ways.

```
o=- 2890844526 2890844526 IN IP4 host.example.com
c=IN IP4 10.0.1.1

a=group:KUMQUAT b c1 c2 c3 c4

m=audio 9 RTP/AVP 0 8 97
c=IN IP4 0.0.0.0
a=mid:c1
a=label:speaker-audio
a=rtcp-mux
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
```

Note that different constituent MDs can use the same payload types (for the same or different codecs), because the kumquat encapsulation captures the constituent MD ordinal index separately from the payload type.

```
m=audio 9 RTP/AVP 0 8 97
c=IN IP4 0.0.0.0
a=mid:c2
a=label:floor-mic
a=rtcp-mux
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 G722

m=video 9 RTP/AVP 103 104
c=IN IP4 0.0.0.0
a=mid:c3
a=label:speaker-video
a=rtcp-mux
a=rtpmap:103 H261/90000
a=rtpmap:104 MPV/90000

m=video 9 RTP/AVP 103 104
c=IN IP4 0.0.0.0
a=mid:c4
a=label:slides
```

```
a=rtcp-mux
a=rtpmap:103 H261/90000
a=rtpmap:104 MPV/90000

m=multipart 10000 RTP/AVP 127
a=mid:b
a=rtcp-mux
a=rtpmap:127 kumquat
a=candidate:0 1 UDP 2113601791 10.0.1.1 10000 typ host
a=candidate:1 1 UDP 1694194431 198.51.100.32 51000 typ srflx \
    raddr 10.0.1.1 rport 10000
```

4.3. Virtual Classroom with One Audio Stream, Two Video Streams, and a Group of Video Streams

This example is the teacher's connection to a virtual classroom server. The media descriptions are tagged using the "content" attribute. [RFC4796] The media comprises:

1. one audio channel, for sending the teacher's voice and receiving the voice of a selected student

2. one video channel, for sending the teacher's presentation

3. one video channel, for sending the teacher's face

4. one video channel, for receiving a dynamically varying set of students' faces

The fourth TF (for students' faces) contains a large and variable set of video captures. These can be handled by a single TF because they all have essentially similar roles -- the application will process them as a set. As Adam Roach would say, "no control surfaces are necessary to talk about and/or manipulate the individual streams". In particular, this allows a large number of captures to be handled without mentioning them in the SDP, at the expense of not allowing the SDP to describe any of them individually. Similarly, the number of captures can vary without having to renegotiate the SDP.

(In contrast, the third TF (the teacher's face) is a separate TF because it is processed in a different role than that of the students' faces.)

In unbundled usage, there would be one transport association for the fourth TF. Incoming RTP from that association would be demultiplexed by the application based on the SSRC values, which would be unique for each student. With bundling, once the single transport TF is

demultiplexed based on the ordinal index in the kumquat encapsulation, deencapsulated RTP packets destined for the fourth TF (index = 3) would be further demultiplexed by their SSRC values.

The offered SDP is:

```
        o=- 2890844526 2890844526 IN IP4 host.example.com
        c=IN IP4 10.0.1.1
        a=group:KUMQUAT b c1 c2 c3 c4
```

The audio channel is send/receive.

```
        m=audio 9 RTP/AVP 0 8 97
        c=IN IP4 0.0.0.0
        a=mid:c1
        a=label:speaker-audio
        a=content:speaker
        a=rtcp-mux
        a=rtpmap:0 PCMU/8000
        a=rtpmap:8 PCMA/8000
        a=rtpmap:97 iLBC/8000
```

The teacher's face and presentation are send-only.

```
        m=video 9 RTP/AVP 103 104
        c=IN IP4 0.0.0.0
        a=mid:c2
        a=label:speaker-video
        a=content:speaker
        a=sendonly
        a=rtcp-mux
        a=rtpmap:103 H261/90000
        a=rtpmap:104 MPV/90000

        m=video 9 RTP/AVP 105 106
        c=IN IP4 0.0.0.0
        a=mid:c3
        a=label:presentation
        a=content:slides
        a=sendonly
        a=rtcp-mux
        a=rtpmap:105 H261/90000
        a=rtpmap:106 MPV/90000
```

The student video input is receive-only and is limited to 24 simultaneous SSRCs.

```
        m=video 9 RTP/AVP 105 106
```

```
      c=IN IP4 0.0.0.0
      a=mid:c4
      a=label:student-thumbnails
      a=recvonly
      a=max-recv-ssrc:* 24
      a=rtcp-mux
      a=rtpmap:105 H261/90000
      a=rtpmap:106 MPV/90000

      m=multipart 10000 RTP/AVP
      a=mid:b
      a=rtcp-mux
      a=candidate:0 1 UDP 2113601791 10.0.1.1 10000 typ host
      a=candidate:1 1 UDP 1694194431 198.51.100.32 51000 typ srflx \
          raddr 10.0.1.1 rport 10000
```

5. Syntax and Semantics

TBD (Here lies the real description.)

5.1. Constructing a Session Description

TBD

5.2. Constructing an Answer

TBD

5.3. Offer/Answer Considerations

TBD

5.4. Multiplexing and Demultiplexing Media Streams

SDP bundling uses a payload type named "kumquat" to encapsulate the RTP packets of several constituent TFs into RTP packets of one TF. Each constituent TF has a distinct index value in the range 0 to 254 (inclusive). When kumquat is used within SDP bundling, the index value is the ordinal index of the MD within the session description. (The indexes start with 0 for the first MD.)

When the application delivers a payload (and associated descriptive information such as SSRC) in the context of a constituent MD to be transmitted, it is encapsulated into a kumquat payload and the kumquat payload is transmitted using the transport association of the bundle MD.

When a kumquat payload arrives on the transport association of the bundle MD, the kumquat payload is interpreted to construct a payload (and associated descriptive information). That payload is delivered to the application in the context of the constituent MD identified by the index value.

5.4.1. The "kumquat" Payload Format

The format of a kumquat protocol payload contains a four-octet fixed part followed by zero or more CSRC identifiers, header extension, and the encapsulated payload. Note that this diagram is of the kumquat payload only, and does not include the RTP header before the payload.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |V=2|0|X|  CC   |M|     PT      |   index       |     0         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            contributing source (CSRC) identifiers             |
   |                             ....                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           extension                           |
   |                             ....                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           encapsulated payload                |
   |                             ....                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

V: This field contains the value 2.

0 (bit 2): This field contains the value 0.

X: If this field is 1, the extension field is present.

CC: This field contains the count of the number of CSRC identifiers that follow the fixed part.

M: This field contains the "marker" bit associated with the encapsulated payload.

PT: This field contains the payload type number associated with the encapsulated payload. The meaning of PT is defined by the TF identified by the index field.

index: This field contains the index value identifying the constituent TF that the encapsulated payload is associated with. The range of index values is 0 to 254 (inclusive). The value 255 is reserved for further standardization and MUST NOT be used.

0 (bits 24 to 31): This field is reserved for further standardization. It MUST be set to 0 when the payload is created and MUST be ignored when the payload is interpreted.

contributing source (CSRC) identifiers: This variable-length field contains the four-octet CSRC identifiers associated with the encapsulated payload. The number of CSRC identifiers is given by the CC field.

extension: This variable-length field is present only if the X field is 1. If it is present, its format is the same as the extension field of the RTP header. In particular, its length is always a multiple of four octets.

encapsulated payload: This variable-length field contains the payload of the payload type specified by the PT field (interpreted in the context of the constituent MD identified by the index field).

There is no defined meaning for the RTP marker bit in association with a kumquat payload. (Note that this is the marker bit in the RTP header that precedes the kumquat payload, not the M field of the kumquat payload itself.) Its value MUST be 0.

The kumquat payload represents an RTP packet containing the following data:

V: The V field is 2.

P: The pad field is unspecified, because the need for padding is determined only when the RTP packet is considered in the context of the transport protocol.

X, CC, M, PT: These fields are taken from the corresponding fields of the kumquat payload data.

sequence number, timestamp, SSRC identifier: These fields are taken from the corresponding fields of RTP header before the kumquat payload.

extension, CSRC identifiers: These fields are taken from the corresponding fields of the kumquat payload data.

payload: This field is taken from the encapsulated payload field of the kumquat payload data.

Graphically, the kumquat encoding sets up the following equivalence between an RTP packet of the constituent TF and an RTP packet of the bundle TF:

RTP packet in the context of the bundle media description (with PT1 specifying kumquat encoding):

```
         0                   1                   2                   3
         0 1 2 3  4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     RTP header:
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |V=2|P|X1|  0    |0|     PT1     |       sequence number         |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                            timestamp                           |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |            synchronization source (SSRC) identifier            |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                            extension (per X1 bit)              |
        |                             ....                               |
        +=+=+=+==+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
     Payload of kumquat payload type:
        +=+=+=+==+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
        |V=2|0|X2|  CC   |M|     PT2     |   index       |     0         |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                            extension (per X2 bit)              |
        |                             ....                               |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |             contributing source (CSRC) identifiers (per CC)    |
        |                             ....                               |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                            encapsulated payload                |
        |                             ....                               |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

RTP packet in the context of the constituent media description identified by index:

```
         0                   1                   2                   3
         0 1 2 3  4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     RTP header:
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |V=2|P|X2|  CC   |M|     PT2     |       sequence number         |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                            timestamp                           |
```

```
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+
        |            synchronization source (SSRC) identifier             |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+
        |                            extension (per X2 bit)               |
        |                              ....                               |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+
        |             contributing source (CSRC) identifiers (per CC)     |
        |                              ....                               |
        +=+=+=+==+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+==+
     Payload of PT2 payload type:
        +=+=+=+==+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+==+
        |                            encapsulated payload                 |
        |                              ....                               |
        +-+-+-+--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+
```

The kumquat encapsulation usually adds four octets to the length of the encapsulated RTP packet. The encapsulation overhead can be larger if there is a need for a separate RTP header extension for the kumquat RTP packet.

5.5. RTCP, SSRC, and RTP Sessions

TBD

5.6. ICE considerations

TBD

6. Compatibility Considerations

6.1. Backward Compatibility during Offer/Answer

TBD

6.2. Backward Compatibility with Existing Devices

TBD

7. Design Features and Comparison

Key:

```
        x = feature present in proposal
        - = feature not present in proposal
        . = feature not discussed in proposal
      N/A = feature is not relevant because of another feature choice
```

	worley-sdp-bundle-03 (KUMQUAT)	ietf-mmusic-sdp-bundle-negotiation-03 (BUNDLE)	holmberg-mmusic-sdp-mmt-negotiation-00 (MMT)	alvestrand-one-rtp-02 (TOGETHER)	ejzak-mmusic-bundle-alternatives-00	Roach alternative 1a (roach-mmusic-mlines-00)	Roach alternative 1b	Roach alternative 2	westerlund-avtcore-transport-multiplexing-04 (SHIM)
MD grouping:									
one	-	-	-	-	-	-	x	-	-
per type	-	-	-	-	-	x	-	-	-
none	x	x	x	x	x	-	-	x	x
Separate bundle MD:									
no	-	x	-	x	x	x	x	x	x
m=anymedia	-	-	x	-	-	-	-	-	-
m=audio	x	-	-	-	-	-	-	-	-
m=multipart	-	-	-	-	-	-	-	-	-
Ports in constituent MDs in offer:									
N/A	-	-	-	-	-	x	x	-	-
same	-	-	-	x	x	-	-	x	x
different	-	x	x	-	x	-	-	-	-
null	x	-	-	-	-	-	-	-	-
Immediate update:									
none	-	-	x	x	x	x	x	x	x
for support	-	x	-	-	-	-	-	-	-
for compat.	x	-	-	-	-	-	-	-	-
Constituent MD ports after establishment:									
N/A	-	-	-	-	-	x	x	-	-
same	-	x	-	x	-	-	-	x	x
different	-	-	-	-	-	-	-	-	-
null	-	-	-	-	-	-	-	-	-
rejected	x	-	x	-	x	-	-	-	-
Bundle MD payload types:									
N/A	-	-	-	-	-	x	x	-	-
one MD	-	x	-	x	.	-	-	x	x
all MDs	-	-	x	-	.	-	-	-	-
encap.	x	-	-	-	.	-	-	-	-

Constituent MD payload types:									
N/A	-	-	-	-	-	x	x	-	-
overlapping	x	-	-	-	.	-	-	x	x
distinct	-	x	x	x	.	-	-	-	-
Demultiplexing based on:									
N/A	-	-	-	-	-	x	x	-	-
PT	-	x	x	x	.	-	-	-	-
encap.	x	-	-	-	.	-	-	x	x
Rejection of bundle MD based on:									
N/A	-	x	-	x	x	x	x	x	x
media type	-	-	x	-	-	-	-	-	-
codec	x	-	-	-	-	-	-	-	-

7.1. Aggregation of Constituent Media Descriptions

Are the constituent media descriptions combined into grouped media descriptions?

- All media are combined into one media description.

- All media of each single type are combined into one media description (with that type).

- Each constituent media description is separate in the session description.

7.2. Presence of a Bundle Media Description and Its Media Type

Is there a separate bundle media description, and if so, what media type does it have?

- There is no separate bundle media description.

- There is a separate bundle media description of type "anymedia".

- There is a separate bundle media description of type "audio".

- There is a separate bundle media description of type "multipart".

7.3. Ports in Constituent Media Descriptions in Offer

What are the port numbers in the constituent media descriptions in an offer?

- There are not multiple media descriptions because constituent MDs are combined.

- All constituent MDs have the same port number.

- Constituent MDs have different port numbers.

7.4. Immediate Update

Is an immediate updated offer/answer used during session establishment?

- No.

- Yes, when establishing a session using bundling.

- Yes, when establishing a session not using bundling.

7.5. Effective Media Description Ports after Session Establishment

What are the effective port numbers in MDs after the session is established?

- There are not multiple media descriptions because constituent MDs are combined.

- Port numbers in the MDs are the same.

- Port numbers in the MDs are different.

- All but one MD have a null addresses.

- All but one MD have a zero port number (and thus are formally rejected).

7.6. Payload Types in the Bundled Media Description

What payload types are listed for the bundled MD?

- There is no MD describing the bundle as a whole.

- The bundle MD lists the payload types of one constituent MD.

- The bundle MD lists the payload types of all constituent MDs.

- The bundle MD lists one payload type for an encapsulation codec.

7.7. Relationship of Payload Types of Constituent Media Descriptions

What is the relationship between the payload types of the constituent MDs?

- There are not multiple media descriptions because constituent MDs are combined.

- Different constituent MDs may have overlapping payload type numbers.

- Different constituent MDs may not have overlapping payload type numbers.

7.8. Basis of Demultiplexing

What is the basis for the demultiplexing of RTP?

- Demultiplexing is not done because incoming RTP is not attributed to specific constituent MDs (possibly because constituent MDs are combined).

- Demultiplexing is done based on payload type numbers.

- Demultiplexing is done based on data carried in an encapsulation.

8. Demultiplexing Considerations

This section discusses the constraints regarding demultiplexing datagrams from multiple protocols that are presented on one transport flow. This is an expansion of the analysis in [RFC5764] section 5.1.2.

The first octets of datagrams generated by particular protocols are:

Protocol	First octet	Second octet	Third octet	Fourth octet
STUN	0x00, 0x01	0x00, 0x01		
RTP	0xB0 to 0xBF	0x00 to 0xC7, 0xCD to 0xFF		
RTCP	0xB0 to 0xBF	0xC8 to 0xCC		
DTLS	0x14 to	0x03	0x03	

```
   |            | 0x17       |                |           |            |
   |            |            |                |           |            |
   | SCTP       | source     | source port    | dest.     | dest. port |
   |            | port high  | low            | port high | low        |
   +------------+------------+----------------+-----------+------------+
```

TBD RFC 5764 specifies that the first octet of a DTLS packet is in the range 0x14 to 0x3F. RFC 5246 and RFC 6374 together specify the first octet is a "ContentType", with the range 0x14 to 0x17. Are additional octet values reserved for expansion? What is the range that should be reserved in practice?

The most generalized stack of protocols we consider is this:

```
   Layer 5:  ... application interfaces ...
              |||      |||     |||     |||
               V        V       V       V
               |        |       |       |
               |        |       |       |
   Layer 4:    |        |       |      SCTP
               |        |       |       |
               |        |       |       |
              RTP     SRTP      |       |
   Layer 3:   RTCP    SRTCP    SCTP    DTLS   [ STUN ]
                \       |       |       |        /
                 --------------- ----------------
                                V
                                |
                                |
   Layer 2:                  [ DTLS               STUN ]
                             [   \                 /   ]
                             [    ------- ---------    ]
                             [           V             ]
                                         |
                                         |
   Layer 1:                           [ TURN ]
                                         |
                                         |
   Layer 0:                             UDP
```

TBD Do we consider encryption of the bundle transport flow itself? If not, why not, as it seems to be simpler (and slightly more secure) than encrypting each multiplexed protocol itself.

Layer 0: UDP This is the base layer of this analysis, where packets are carried by UDP.

Layer 1: TURN (optional) If a packet arrives from a TURN relay for which we are a client, the TURN encapsulation must be removed before further processing. This need can be detected because the packet arrives from the client-facing address/port of a TURN server of which this entity is a client.

Layer 2: DTLS applied to the bundle transport flow
(optional)
If DTLS is applied to the bundle transport flow as a whole, that use of DTLS will have been negotiated. However, before deciphering, STUN packets need to be separated. STUN packets can be distinguished from DTLS packets by their first or second octets.

Layer 3: Core demultiplexing At this layer, most of the protocols are demultiplexed. RTP/SRTP, SRTP/SRTCP, DTLS, and STUN are distinguished by the values of the first two octets of the packet. SRTP/SRTCP is distinguished from RTP/RTCP by negotiation with the other endpoint -- SRTP/SRTCP is never multiplexed with RTP/RTCP.

SCTP can be distinguished from the other protocols if the other endpoint agrees to use only SCTP port numbers 0xC000 and higher, that is, within the dynamic/private port range for SCTP [RFC4960]. This is specified, because this demultiplexing is only done when both endpoints support bundling.

Layer 4: SCTP over DTLS If the layer 3 protocol is DTLS, the protocol above it will always be SCTP.

Layer 5: Application interface demultiplexing The method used to demultiplex the various application interface streams varies depending whether encapsulation is used and if not, on the layer 3 /4 protocol. If the layer 3 protocol is encapsulated, the encapsulation determines which constituent media description is to be assigned to this packet, and then application demultiplexing is done as normal for for that particular media description. Otherwise, the constituent media description must be determined by a method that is specific to the layer 3/4 protocol:

RTP/SRTP An RTP/SRTP packet is assigned to the constituent media description which specifies its payload type number. (If encapsulation is not used, the constituent media descriptions must specify distinct payload type numbers.)

RTCP/SRTCP An RTCP/SRTCP sub-packet is dispatched to the application which receives the RTP media stream containing the SSRC that is carried in the RTCP sub-packet.

SCTP An SCTP packet is assigned to the constituent media description which specifies its destination port number. (If encapsulation is not used, the constituent media descriptions must specify distinct SCTP port numbers.)

9. Security Considerations

If an SBC wishes to prevent positively the transport of certain media types or codecs, and enforces that by examining the content of RTP packets, the use of kumquat encoding may defeat the examination.

TBD

10. IANA Considerations

TBD

11. Acknowledgments

Many people have provided input for this proposal regarding both the technical aspects and the organization of the presentation. Chief among them are the authors of the predecessor proposals ([I-D.alvestrand-one-rtp] (TOGETHER), [I-D.holmberg-mmusic-sdp-mmt-negotiation] (MMT), and [I-D.ietf-mmusic-sdp-bundle-negotiation] (BUNDLE)): Harald Alvestrand, Jonathan Lennox, and Christer Holmberg. In addition, input was provided by Charles Eckel, Andrew Hutton, Cullen Jennings, Hadriel Kaplan, Paul Kyzivat, Adam Roach, and Robert Sparks.

12. Revision History

Note to RFC Editor: Please remove this section before publication.

12.1. draft-worley-sdp-bundle-00

Initial version.

12.2. Changes from draft-worley-sdp-bundle-00 to draft-worley-sdp-bundle-01

Thoroughly revise the text and structure of the document.

12.3. Changes from draft-worley-sdp-bundle-01 to draft-worley-sdp-bundle-02

Heavily revise Terminology regarding media flows.

Revise Desiderata, including adding that multiple separate bundles must be possible, and noninterference with ICE negotiation.

Add section on ICE considerations.

Change "fusion" to "bundle".

Use a=rtcp-mux in examples to be more realistic (and to shorten the examples).

Correct the use of ICE in answers; ICE candidates are not provided if an offered MD does not contain ICE candidates.

Add an example of a fast-start offer.

12.4. Changes from draft-worley-sdp-bundle-02 to draft-worley-sdp-bundle-03

Add design comparison Section 7.

Use bibxml references.

Add DES C9, regarding continued usage of transcoding facilities offered by intermediate entities.

12.5. Changes from draft-worley-sdp-bundle-03 to draft-worley-sdp-bundle-04

Add demultiplexing considerations Section 8.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

13.2. Informative References

[RFC2327] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[RFC3725] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[I-D.alvestrand-mmusic-msid]
Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol", draft-alvestrand-mmusic-msid-02 (work in progress), December 2012.

[I-D.alvestrand-one-rtp]
Alvestrand, H., "SDP Grouping for Single RTP Sessions", draft-alvestrand-one-rtp-02 (work in progress), September 2011.

[I-D.holmberg-mmusic-sdp-mmt-negotiation]

Holmberg, C., Alvestrand, H., and J. Lennox, "Multiplexed Media Types (MMT) Using Session Description Protocol (SDP) Port Numbers", draft-holmberg-mmusic-sdp-mmt-negotiation-00 (work in progress), October 2012.

[I-D.ietf-avtcore-multi-media-rtp-session]
Westerlund, M., Perkins, C., and J. Lennox, "Multiple Media Types in an RTP Session", draft-ietf-avtcore-multi-media-rtp-session-01 (work in progress), October 2012.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-03 (work in progress), February 2013.

[I-D.jennings-mmusic-media-req]
Jennings, C., Uberti, J., and E. Rescorla, "Requirements from various WG for MMUSIC", draft-jennings-mmusic-media-req-00 (work in progress), February 2013.

[I-D.worley-service-example]
Worley, D., "Session Initiation Protocol Service Example -- Music on Hold", draft-worley-service-example-11 (work in progress), February 2013.

Author's Address

Dale R. Worley
Ariadne Internet Services, Inc.
738 Main St.
Waltham, MA 02451
US

Phone: +1 781 647 9199
Email: worley@ariadne.com