Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 08, 2013

A. B. Roach
Mozilla
M. Thomson
Microsoft

May 07, 2013

Using SDP with Large Numbers of Media Flows
draft-roach-rtcweb-plan-a-00

Abstract

A recurrent theme in WebRTC has been the need to handle very large numbers of media flows. Unfortunately, naive uses of SDP do not handle this case particularly well. This document describes a modest set of extensions to SDP which allow it to cleanly handle arbitrary numbers of flows while still retaining a large degree of backward compatibility with existing and non-RTCWEB endpoints.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on November 08, 2013.

Copyright Notice

Table of Contents

1. Introduction

A recurrent theme in WebRTC has been the need to cleanly handle very large numbers of media flows. For instance, a video conferencing application might have a main display plus thumbnails for 10 or more other speakers all displayed at the same time. If each video source is encoded in multiple resolutions (e.g., simulcast or layered coding) and also has FEC or RTX, this could easily add up to 30 or more independent RTP flows.

The standard way of encoding this information in SDP is to have each RTP flow (i.e., SSRC) appear on its own m-line. For instance, the SDP for two cameras with audio from a device with a public IP address

could look something like:

```
v=0
o=- 20518 0 IN IP4 203.0.113.1
s=
```



```
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 54400 RTP/SAVPF 0 96
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.1 54400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.1 54401 typ host

m=video 55400 RTP/SAVPF 97 98
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.1 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.1 55401 typ host

m=video 56400 RTP/SAVPF 99 100
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:100 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.1 56400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.1 56401 typ host
```

Unfortunately, as the number of independent media sources starts to increase, the scaling properties of this approach become problematic. In particular, SDP currently requires that each m-line have its own transport parameters (port, ICE candidates, etc.), which can get

expensive. For instance, the [RFC5245] pacing algorithm requires that new STUN transactions be started no more frequently than 20 ms; with 30 RTP flows, which would add 600 ms of latency for candidate gathering alone. Moreover, having 30 persistent flows might lead to excessive consumption of NAT binding resources.

A related issue is the number of payload types. Even multiple sources are multiplexed over the same transport flow they must somehow be demultiplexed. Consider the case where we want to be able to transmit 32 video thumbnails (this is large, but not insane). In the model described above, each of these flows would need its own m-line and its own set of codecs. If each side supports three

separate codecs (e.g., H.263, H.264, VP8, and VP9), then we have just consumed 128 payload types, which exceeds the available dynamic payload space. This makes demuxing on payload type problematic in some cases.

This document specifies a small number of modest extensions to SDP which are intended to reduce the transport impact of using a large number of flows. The general design philosophy is to maintain the existing SDP negotiation model while simply reducing the consumption of network resources.

2. Terminology

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

This draft uses the API and terminology described in [webrtc-api].

5-tuple: A collection of the following values: source IP address, source transport port, destination IP address, destination transport port and transport protocol.

Transport-Flow: An transport 5 Tuple representing the UDP source and destination IP address and port over which RTP is flowing.

PC-Track: A source of media (audio and/or video) that is contained in a PC-Stream. A PC-Track represents content comprising one or more PC-Channels.

m-line: An SDP [RFC4566] media description identifier that starts with an "m=" field and conveys the following values: media type, transport port, transport protocol and media format descriptions.

Offer: An [RFC3264] SDP message generated by the participant who wishes to initiate a multimedia communication session. An Offer describes the participant's capabilities for engaging in a multimedia session.

Answer: An [RFC3264] SDP message generated by the participant in response to an Offer. An Answer describes the participant's capabilities in continuing with the multimedia session with in the constraints of the Offer.

This draft avoids using terms that implementors do not have a clear idea of exactly what they are - for example RTP Session.

3. Syntax Conventions

The SDP examples given in this document deviate from actual on-the-wire SDP notation in several ways. This is done to facilitate readability and to conform to the restrictions imposed by the RFC formatting rules. These deviations are as follows:

- Any line that is indented (compared to the initial line in the SDP block) is a continuation of the preceding line. The line break and indent are to be interpreted as a single space character.

- Empty lines in any SDP example are inserted to make functional divisions in the SDP clearer, and are not actually part of the SDP syntax.

- Excepting the above two conventions, line endings are to be interpreted as <CR><LF> pairs (that is, an ASCII 13 followed by an ASCII 10).

- Any text starting with the string "//" to the end of the line is inserted for the benefit of the reader, and is not actually part of the SDP syntax.

4. Requirements

This document is intended to address the following requirements, based on those from [I-D.jennings-mmusic-media-req].

1. Support many media flows but minimize the number of transport flows.

   This requirement is partly satisfied by BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation]; however, BUNDLE still requires a large number of ports and ICE candidates in the initial offer. This can create serious latency issues, as described in Section 1. The mechanisms in Section Section 6.1 of this document address those issues.

3. Be able to successfully negotiate media with both legacy SIP devices and new devices (whether SIP or RTCWEB) with a single offer/answer exchange. If both endpoints support multiplexed media, then multiplexing should be negotiated. Otherwise, non-multiplexed media should be used.

   The interaction of this mechanism with non-WEBRTC devices is described in Section 6.5.

5. Provide a mechanism for harmonizing flow parameters for different m-lines when they are multiplexed over the same transport.

   [I-D.nandakumar-mmusic-sdp-mux-attributes] documents the required procedures.

7. Allow different sources (e.g., cameras) to use different codecs. For example, if one camera had hardware encoders for VP8 while another had encoders for H.264, the device may wish to negotiate different codecs.

   This requirement is also already satisfied by existing SDP mechanisms; we simply need to preserve them.

9. Be able to independently set parameters such as resolution and bandwidth, independently for each PC-Track, preferably even when they are all multiplexed over the same transport flow.

   Section 6.2 of this document satisfies the multiplexing requirement and the normal SDP mechanisms are used for parameters.

11. Be able to identify the PC-Tracks with an identifier that is stable over the duration of the session.

   Section 6.2 of this document explains track identification.

Note that this document does not attempt to address the issue of adding a stream with little or no chance of glare. See [I-D.roach-rtcweb-glareless-add] for the description of a technique that can be applied to any SDP offer/answer session establishment protocol to eliminate mid-session glare.

5. Overview

This section provides an overview of the approach specified in this document.

- We retain the existing SDP model that the m-line is the basic unit of media negotiation/representation. Each independent unit (i.e., a specific encoding of a PC-Track) is represented on its own m-line.

- BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] is used to multiplex multiple m-lines (and their corresponding media) onto the same set of transport flows.

- Both RTP payload type (PT) and SSRC are used to de-multiplex flows multiplexed via bundle. This allows for many more flows to be bundled than the limited number of PTs available.

- In order to minimize the number of transport parameters that need to be allocated during the gathering phase, m-lines can be tagged as "bundle only". Such m-lines in an offer will be ignored by legacy endpoints but can be negotiated by endpoints that support

the mechanisms specified in this document.

6. Detailed Description

6.1. Bundle-Only M-Lines

As discussed in Section 1, even with bundle, it is expensive to allocate ICE candidates for a large number of m-lines. An offer can contain "bundle-only" m-lines which will be negotiated only by endpoints which implement this specification and ignored by other endpoints.

In order to offer such an m-line, the offerer does two things:

- Sets the port in the m-line to 0. This indicates to old endpoints that the m-line is not to be negotiated.
- Adds an a=bundle-only line. This indicates to new endpoints that the m-line is to be negotiated if (and only if) bundling is used.

An example offer that uses this feature looks like this:

```
v=0
o=- 20518 0 IN IP4 203.0.113.1
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 54400 RTP/SAVPF 0 96
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=candidate:0 1 UDP 2113667327 203.0.113.1 54400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.1 54401 typ host
```



```
m=video 0 RTP/SAVPF 97 98
```

```
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only

m=video 0 RTP/SAVPF 99 100
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:100 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
```

An old endpoint simply rejects the bundle-only m-lines by responding with a 0 port. (This isn't a normative statement, just a description of the way the older endpoints are expected to act.)

```
v=0
o=- 20518 0 IN IP4 203.0.113.1
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 55400 RTP/SAVPF 0 96
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

m=video 0 RTP/SAVPF 97 98
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv

m=video 0 RTP/SAVPF 99 100
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
```

```
a=rtpmap:100 VP8/90000
a=sendrecv
```

A new endpoint accepts the m-lines (both bundle-only and regular) by offering m-lines with a valid port, though this port may be duplicated as specified in Section 6 of [I-D.ietf-mmusic-sdp-bundle-negotiation]. For instance:

```
v=0
o=- 20518 0 IN IP4 203.0.113.2
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 55400 RTP/SAVPF 0 96
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host

m=video 55400 RTP/SAVPF 97 98
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only

m=video 55400 RTP/SAVPF 99 100
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:100 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
```

Endpoints MUST NOT accept bundle-only m-lines if they are not part of an accepted bundle group.

6.2. Flow Demultiplexing

As noted above, if a large number of m-lines are used and each codec in each m-line uses its own PT, it is possible to exceed the number of possible PT values. This makes PT-only demultiplexing insufficient in some cases.

Offerers conformant to this specification MUST do one of the following:

- Use non-overlapping PT values for each m-line in any given bundle group.
- Provide distinct a=ssrc attributes for each m-line which uses overlapping PT values with any other m-line. [Technically, this is a general case of the previous point.]

If the offerer uses overlapping PT values for any two m-lines in a given bundle group, the answerer MUST supply distinct a=ssrc attributes for those m-lines.

Upon receipt of an RTP datagram on a port that is being used with multiplexing, implementors SHOULD follow a procedure equivalent to the following to demultiplex streams:

- If the SSRC in the received packet matches one that has been mapped to an m-line (e.g., via a=ssrc attributes), then the packet corresponds to that m-line.
- If the SSRC in the received packet does not matches one that has associated with an m-line, but the PT value appears on only one m-line, then the packet corresponds to the m-line containing that PT.
- Otherwise, discard the packet.

Note that this approach means that if PT values overlap between two m-lines, then those m-lines cannot be demultiplexed prior to receiving the m-line-to-ssrc mapping (e.g., in the SDP answer). For

instance, if the offerer wants two m-lines to be rendered prior to receipt of the SDP answer, it can use non-overlapping PT values on those m-lines.

6.3. Indicating Simulcast Groups

Simulcast refers to taking a single capture (e.g., a camera), and encoding it multiple times at different resolutions and / or frame rates. For example, a device with a single HD camera may send one version of the video at full HD resolution, and a second version encoded at a low resolution. This would allow a video conferencing bridge to be able to send the high resolution copy to some destination and low resolution copy to other destinations without having to recode the video at the conference bridge.

This document proposes that simulcast be done by defining a new SDP group [RFC5888] called SIMULCAST. Any m-lines that are in the same SIMULCAST group are alternative encodings of the same media capture.

One of the advantages of this approach is it works well with the many existing RTP definitions that have been done in the past as well as others that may be done in the future.

The order of m-lines in a SIMULCAST group determines the relative size of the encoded streams. Streams at lower quality appear before streams of higher quality. The entity creating the session description can choose to order m-lines based on any quality criteria (resolution, framerate, sample rate), but they SHOULD choose an ordering that places streams with a lower average bitrate before higher bitrate streams.

Providing an order to SIMULCAST groupings allows an intermediary (such as a Media Translator [RFC5117]) to be able to select an appropriate SIMULCAST layer without inspecting the media stream, which could otherwise require decrypting and possibly partially decoding media packets.

6.4. Identifying Flows

While this topic is largely out of scope for SDP, the SSRC value can be used as a flow identifier. One minor caveat with this approach is the ability to deal with the SSRC collision resolution procedure described in section 8.2 of [RFC3550]. In the rare circumstances that such an SSRC change is required, then any party that has changed its SSRC needs to inform the remote participants of the updated mapping, e.g. via a new SDP offer. In WebRTC use cases, this would trigger an onrennegotiationneeded event.

6.5. Compatibility with Non-RTCWEB uses

Due to the fact that this approach re-uses existing SDP constructs for indicating parameters in a media section, it remains compatible

with non-RTCWEB clients. Of particular note is the handling of "bundle-only" media sections, described in Section 6.1. Offers generated by an RTCWEB client and sent to a non-RTCWEB client will simply negotiate those media the RTCWEB client did not use the "bundle-only" extension with. This allows RTCWEB clients to select which media streams are important for interoperability with non-RTCWEB clients (by not making them bundle-only), and which ones are not. Offers generated by non-RTCWEB clients will simply omit any bundle-related attributes, and the RTCWEB client will be able to process the SDP otherwise identically to the SDP received from RTCWEB clients: each m-line represents a different media stream, and contains a description of that stream in a syntax identical to the syntax used between RTCWEB clients.

With the bundle-only approach, only those streams that are "important for interoperability" will require allocation of ports and ICE exchanges. By doing so, working with non-multiplexing clients is enabled without requiring excess resource allocation for those streams that are not critical for proper user experience.

Aside from BUNDLE, the bundle-only mechanism, and the rules around port demultiplexing, this proposal requires no additional extensions to SDP or the offer/answer model.

7. Examples

In all of these examples, there are many lines that are wrapped due to column width limitation. It should be understood these lines are not wrapped in the real SDP.

The convention used for IP addresses in this drafts is that private IP behind a NAT come from 192.0.2.0/24, the public side of a NAT comes from 198.51.100.0/24 and the TURN servers have addresses from 203.0.113.0/24. Typically the offer has an IP ending in .1 and the answer has an IP ending in .2.

The examples do not include all the parts of SDP that are used in RTCWeb (See [I-D.ietf-rtcweb-rtp-usage]) as that makes the example unwieldy to read but instead focuses on showing the parts that are key for the multiplexing.

7.1. Simple example with one audio and one video

The following SDP shows an offer that offers one audio stream and one video steam with both a STUN and TURN address.

```
v=0
o=- 20518 0 IN IP4 198.51.100.1
```



```
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:074c6550
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068
a=fingerprint:sha-1
        99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07
a=group:BUNDLE m1 m2

m=audio 56600 RTP/SAVPF 0 109
a=mid:m1
a=rtpmap:0 PCMU/8000
a=rtpmap:109 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=candidate:0 1 UDP 2113667327 192.0.2.1 54400 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.1 54401 typ host
```

```
a=candidate:0 1 UDP 694302207 198.51.100.1 55500 typ srflx raddr
        192.0.2.1 rport 54400
a=candidate:1 2 UDP 169430220 198.51.100.1 55501 typ srflx raddr
        192.0.2.1 rport 54401
a=candidate:0 1 UDP 73545215 203.0.113.1 56600 typ relay raddr
        192.0.2.1 rport 54400
a=candidate:1 2 UDP 51989708 203.0.113.1 56601 typ relay raddr
        192.0.2.1 rport 54401

m=video 56602 RTP/SAVPF 99 120
a=mid:m2
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:120 VP8/90000
a=sendrecv
a=rtcp-mux
a=candidate:3 1 UDP 2113667327 192.0.2.1 54402 typ host
a=candidate:4 2 UDP 2113667326 192.0.2.1 54403 typ host
a=candidate:3 1 UDP 694302207 198.51.100.1 55502 typ srflx raddr
        192.0.2.1 rport 54402
a=candidate:4 2 UDP 169430220 198.51.100.1 55503 typ srflx raddr
        192.0.2.1 rport 54403
a=candidate:3 1 UDP 73545215 203.0.113.1 56602 typ relay raddr
        192.0.2.1 rport 54402
a=candidate:4 2 UDP 51989708 203.0.113.1 56603 typ relay raddr
        192.0.2.1 rport 54403
```

The following shows and answer to the above offer from a device that does not support bundle or rtcp-mux.


```
v=0
o=- 16833 0 IN IP4 198.51.100.2
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 60600 RTP/SAVPF 109
```

```
a=rtpmap:109 opus/48000
a=ptime:20
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60400 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.2 60401 typ host
a=candidate:0 1 UDP 1694302207 198.51.100.2 60500 typ srflx raddr
        192.0.2.2 rport 60400
a=candidate:1 2 UDP 1694302206 198.51.100.2 60501 typ srflx raddr
        192.0.2.2 rport 60401
a=candidate:0 1 UDP 73545215 203.0.113.2 60600 typ relay raddr
        192.0.2.1 rport 60400
a=candidate:1 2 UDP 51989708 203.0.113.2 60601 typ relay raddr
        192.0.2.1 rport 60401

m=video 60602 RTP/SAVPF 99
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60402 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60403 typ host
a=candidate:2 1 UDP 694302207 198.51.100.2 60502 typ srflx raddr
        192.0.2.2 rport 60402
a=candidate:3 2 UDP 169430220 198.51.100.2 60503 typ srflx raddr
        192.0.2.2 rport 60403
a=candidate:2 1 UDP 73545215 203.0.113.2 60602 typ relay raddr
        192.0.2.2 rport 60402
a=candidate:3 2 UDP 51989708 203.0.113.2 60603 typ relay raddr
        192.0.2.2 rport 60403
```

The following shows and answer to the above offer from a device that does support bundle.

```
v=0
o=- 16833 0 IN IP4 198.51.100.2
s=
t=0 0
```



```
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
```

```
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 60600 RTP/SAVPF 109
a=rtpmap:109 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=candidate:0 1 UDP 2113667327 192.0.2.2 60400 typ host
a=candidate:0 1 UDP 1694302207 198.51.100.2 60500 typ srflx raddr
        192.0.2.2 rport 60400
a=candidate:0 1 UDP 73545215 203.0.113.2  60600 typ relay raddr
        192.0.2.1 rport 60400

m=video 60600 RTP/SAVPF 99
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=sendrecv
a=rtcp-mux
a=candidate:3 1 UDP 2113667327 192.0.2.2 60400 typ host
a=candidate:3 1 UDP 694302207 198.51.100.2 60500 typ srflx raddr
        192.0.2.2 rport 60400
a=candidate:3 1 UDP 73545215 203.0.113.2  60600 typ relay raddr
        192.0.2.2 rport 60400
```

7.2. Multiple Videos

Simple example showing an offer with one audio stream and two video streams.

```
v=0
o=- 20518 0 IN IP4 198.51.100.1
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7
a=group:BUNDLE m1 m2 m3

m=audio 56600 RTP/SAVPF 0 96
a=mid:m1
a=rtpmap:0 PCMU/8000
```

```
   a=rtpmap:96 opus/48000
   a=ptime:20
   a=sendrecv
   a=rtcp-mux
   a=candidate:0 1 UDP 2113667327 192.0.2.1 54400 typ host
   a=candidate:1 2 UDP 2113667326 192.0.2.1 54401 typ host
   a=candidate:0 1 UDP 694302207 198.51.100.1 55500 typ srflx raddr
           192.0.2.1 rport 54400
   a=candidate:1 2 UDP 169430220 198.51.100.1 55501 typ srflx raddr
           192.0.2.1 rport 54401
   a=candidate:0 1 UDP 73545215 203.0.113.1 56600 typ relay raddr
           192.0.2.1 rport 54400
   a=candidate:1 2 UDP 51989708 203.0.113.1 56601 typ relay raddr
           192.0.2.1 rport 54401

   m=video 56602 RTP/SAVPF 97 98
   a=mid:m2
   a=rtpmap:97 H264/90000
   a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
   a=rtpmap:98 VP8/90000
   a=sendrecv
   a=rtcp-mux
   a=candidate:2 1 UDP 2113667327 192.0.2.1 54402 typ host
   a=candidate:3 2 UDP 2113667326 192.0.2.1 54403 typ host
   a=candidate:2 1 UDP 694302207 198.51.100.1 55502 typ srflx raddr
           192.0.2.1 rport 54402
   a=candidate:3 2 UDP 169430220 198.51.100.1 55503 typ srflx raddr
           192.0.2.1 rport 54403
   a=candidate:2 1 UDP 73545215 203.0.113.1 56602 typ relay raddr
           192.0.2.1 rport 54402
   a=candidate:3 2 UDP 51989708 203.0.113.1 56603 typ relay raddr
           192.0.2.1 rport 54403
   a=ssrc:11111 cname:45:5f:fe:cb:81:e9

   m=video 56604 RTP/SAVPF 99 100
   a=mid:m3
   a=rtpmap:99 H264/90000
   a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
   a=rtpmap:100 VP8/90000
   a=sendrecv
   a=rtcp-mux
   a=candidate:4 1 UDP 2113667327 192.0.2.1 54404 typ host
   a=candidate:5 2 UDP 2113667326 192.0.2.1 54405 typ host
   a=candidate:4 1 UDP 694302207 198.51.100.1 55504 typ srflx raddr
           192.0.2.1 rport 54404
   a=candidate:5 2 UDP 169430220 198.51.100.1 55505 typ srflx raddr
           192.0.2.1 rport 54405
```

```
a=candidate:4 1 UDP 73545215 203.0.113.1 56604 typ relay raddr
```



```
        192.0.2.1 rport 54404
a=candidate:5 2 UDP 51989708 203.0.113.1 56605 typ relay raddr
        192.0.2.1 rport 54405
a=ssrc:22222 cname:45:5f:fe:cb:81:e9
```

7.3. Many Videos

This section adds three video streams and one audio. The video streams are sent in such a way that they they are only accepted if the far side supports bundle using the "bundle only" approach described in Section 6.1. The video streams also use the same payload types so it will not be possible to demux the video streams from each other without using the SSRC values.

```
v=0
o=- 20518 0 IN IP4 198.51.100.1
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7
a=group:BUNDLE m0 m1 m2 m3

m=audio 56600 RTP/SAVPF 0 96
a=mid:m0
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=candidate:0 1 UDP 2113667327 192.0.2.1 54400 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.1 54401 typ host
a=candidate:0 1 UDP 694302207 198.51.100.1 55500 typ srflx raddr
        192.0.2.1 rport 54400
a=candidate:1 2 UDP 169430220 198.51.100.1 55501 typ srflx raddr
        192.0.2.1 rport 54401
a=candidate:0 1 UDP 73545215 203.0.113.1 56600 typ relay raddr
```

```
        192.0.2.1 rport 54400
a=candidate:1 2 UDP 51989708 203.0.113.1 56601 typ relay raddr
        192.0.2.1 rport 54401

m=video 0 RTP/SAVPF 97 98
a=mid:m1
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
```



```
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:11111 cname:45:5f:fe:cb:81:e9

m=video 0 RTP/SAVPF 97 98
a=mid:m2
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:22222 cname:45:5f:fe:cb:81:e9

m=video 0 RTP/SAVPF 97 98
a=mid:m3
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:333333 cname:45:5f:fe:cb:81:e9
```

7.4. Multiple Videos with Simulcast

This section shows an offer with with audio and two video each of which can send it two resolutions as described in Section 6.3. Note that the grouping places the lower bitrate streams first, even though those appear first in the document. All the video is bundle-only.

```
v=0
o=- 20518 0 IN IP4 198.51.100.1
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7
a=group:BUNDLE m0 m1 m2 m3 m4

a=group:SIMULCAST m2 m1
a=group:SIMULCAST m4 m3
```



```
m=audio 56600 RTP/SAVPF 0 96
a=mid:m0
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=candidate:0 1 UDP 2113667327 192.0.2.1 54400 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.1 54401 typ host
a=candidate:0 1 UDP 694302207 198.51.100.1 55500 typ srflx raddr
        192.0.2.1 rport 54400
a=candidate:1 2 UDP 169430220 198.51.100.1 55501 typ srflx raddr
        192.0.2.1 rport 54401
a=candidate:0 1 UDP 73545215 203.0.113.1 56600 typ relay raddr
        192.0.2.1 rport 54400
a=candidate:1 2 UDP 51989708 203.0.113.1 56601 typ relay raddr
        192.0.2.1 rport 54401

m=video 0 RTP/SAVPF 98
b=AS:1500
a=mid:m1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:11111 cname:45:5f:fe:cb:81:e9
```

```
a=framerate:30

m=video 0 RTP/SAVPF 100
b=AS:256
a=mid:m2
a=rtpmap:100 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:22222 cname:45:5f:fe:cb:81:e9
a=framerate:15

m=video 0 RTP/SAVPF 101
b=AS:1500
a=mid:m3
a=rtpmap:101 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:333333 cname:45:5f:fe:cb:81:e9
a=framerate:30
```



```
m=video 0 RTP/SAVPF 103
b=AS:256
a=mid:m4
a=rtpmap:103 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:44444 cname:45:5f:fe:cb:81:e9
a=framerate:15
```

7.5. Video with Simulcast and RTX

This section shows an SDP offer that has an audio and a single video stream. The video stream that is simulcast at two resolutions and has [RFC4588] style re-transmission flows.

```
v=0
```

```
o=- 20518 0 IN IP4 198.51.100.1
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7
a=group:BUNDLE m0 m1 m2 m3 m4

a=group:SIMULCAST m2 m1
a=group:FID m1 m3
a=group:FID m2 m4

m=audio 56600 RTP/SAVPF 0 96
a=mid:m0
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=candidate:0 1 UDP 2113667327 192.0.2.1 54400 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.1 54401 typ host
a=candidate:0 1 UDP 694302207 198.51.100.1 55500 typ srflx raddr
        192.0.2.1 rport 54400
a=candidate:1 2 UDP 169430220 198.51.100.1 55501 typ srflx raddr
        192.0.2.1 rport 54401
a=candidate:0 1 UDP 73545215 203.0.113.1 56600 typ relay raddr
        192.0.2.1 rport 54400
```



```
a=candidate:1 2 UDP 51989708 203.0.113.1 56601 typ relay raddr
        192.0.2.1 rport 54401

m=video 0 RTP/SAVPF 100 // This is the high res video
b=AS:1500
a=mid:m1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=framerate:30
```

```
m=video 0 RTP/SAVPF 101  // This is the low res video
b=AS:256
a=mid:m2
a=rtpmap:100 VP8/90000
a=sendonly
a=rtcp-mux
a=bundle-only
a=framerate:15

m=video 0 RTP/SAVPF 110  // This is the retransmission of high res
b=AS:1500
a=mid:m3
a=rtpmap:101 rtx/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=framerate:30
a=rtcp-fb:101 nack
a=fmtp:101 apt=100;rtx-time=3000

m=video 0 RTP/SAVPF 111  // This is retransmission of low res
b=AS:256
a=mid:m4
a=rtpmap:103 rtx/90000
a=sendonly
a=rtcp-mux
a=bundle-only
a=framerate:15
a=rtcp-fb:111 nack
a=fmtp:11 apt=101;rtx-time=3000
```

8. Security Considerations

TBD

9. IANA Considerations

TBD: registration of SIMULCAST group

10. Acknowledgements

Thanks to Cullen Jennings for his assistance in generating the SDP examples in this document.

Some of the material in this document was taken from [I-D.jennings-rtcweb-plan].

11. References

11.1. Normative References

[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.nandakumar-mmusic-sdp-mux-attributes]
Nandakumar, S., "A Framework for SDP Attributes when Multiplexing", draft-nandakumar-mmusic-sdp-mux-attributes-02 (work in progress), April 2013.

[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.

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

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

11.2. Informative References

[I-D.ietf-rtcweb-rtp-usage]

Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-06 (work in progress), February 2013.

[I-D.jennings-rtcweb-plan]
Jennings, C., "Proposed Plan for Usage of SDP and RTP", draft-jennings-rtcweb-plan-01 (work in progress), February 2013.

[I-D.roach-rtcweb-glareless-add]
Roach, A. B., "An Approach for Adding RTCWEB Media Streams without Glare ", May 2013.

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

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

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

[webrtc-api]
Bergkvist, Burnett, Jennings, Narayanan, , "WebRTC 1.0: Real-time Communication Between Browsers", October 2011.

Available at http://dev.w3.org/2011/webrtc/editor/webrtc.html

Authors' Addresses

Adam Roach
Mozilla
Dallas, TX
US

Phone: +1 650 903 0800 x863
Email: adam@nostrum.com

Martin Thomson
Microsoft
3210 Porter Drive
Palo Alto, CA 94304
US

Phone: +1 650 353 1925
Email: martin.thomson@skype.net