Network Working Group J.
Uberti
Internet-Draft
Google
Intended status: Standards Track May 03,
2013
Expires: November 04, 2013

Plan B: a proposal for signaling multiple media sources in WebRTC.
draft-uberti-rtcweb-plan-00

Abstract

This document explains how multiple media sources can be signaled in WebRTC using SDP, in a fashion that avoids many common problems and provides a simple control surface to the receiver.

Table of Contents

1. Introduction

"Plan B" describes a simple, flexible approach for the negotiation and exchange of multiple media sources (AKA MediaStreamTracks, or MSTs) between two WebRTC endpoints. Each media source can be individually accepted or rejected by the recipient, and preferences on these track requests (e.g. the relative priority of the tracks) can also be provided.

One of the core problems with existing approaches that map media sources to individual m= lines, is that m= lines, by their very nature, specify information about how media should be transported. This means that if N m= lines are created, N media ports (and their corresponding STUN/TURN candidates) need to be allocated. This causes problems when even relatively small numbers of media sources are used, as each source can result in multiple m= lines when techniques like simulcast or RTX are used.

Plan B takes a different approach, and creates a hierarchy within SDP; a m= line defines an "envelope", specifying codec and transport parameters, and [RFC5576] a=ssrc lines are used to describe individual media sources within that envelope. In this manner, Plan B disconnects media streams from transport, and solves the following problems:

- Supports large numbers (>> 100) of sources
- Glareless add/removal of sources
- No need to allocate ports for each source
- Simple binding of MediaStreamTrack to SDP
- Simple use of RTX and FEC streams within a single RTP session
- Impossible to get "out-of-sync" (e.g. getting RTX or FEC flow without its corresponding primary flow)
- Does not require BUNDLE (although BUNDLE can still help)

Plan B mostly uses existing IETF standards, introducing a single new draft [I-D.lennox-mmusic-sdp-source-selection] to specify how individual media sources can be accepted/rejected at the SSRC level via SDP. An extensive review of SDP attributes was performed, in order to determine which SDP attributes that exist at m= line level would need to also exist at "source" level; these attributes are (aside from some trivial exceptions, such as a=label) covered in the aforementioned draft.

Note that at the wire level, RTP is handled identically to Plan A (which specifies separate m= line for each media source), after a successful BUNDLE, in that MSTs are all sent in a single RTP session, and demuxed by SSRC. Therefore, Plan B could be converted to Plan A by a signaling-only gateway (although it would lose the advantages mentioned above).

2. Design Goals

Plan B attempts to provide a comprehensive mechanism for handling large numbers of simultaneous media sources, as might commonly be the case in a video conference or TV guide application. It attempts to address the following concerns:

2.1. Support for a large number of arbitrary sources

In cases such as a video conference, there may be dozens or hundreds of participants, each with their own audio and video sources. A participant may even want to browse conferences before joining one, meaning that there may be cases where there are many such conferences displayed simultaneously.

In these conferences, participants may have varying capabilities and therefore video resolutions. In addition, depending on conference policy, user preference, and the desired UI, participants may be displayed in various layouts, including:

- A single large main speaker with thumbnails for other participants
- Multiple medium-sized main speakers, with or without thumbnails
- Large slides + medium speaker, without thumbnails

These layouts can change dynamically, depending on the conference content and the preferences of the receiver. As such, there are not well-defined 'roles', that could be used to group sources into specific 'large' or 'thumbnail' categories. As such, the requirement Plan B attempts to satisfy is support for sending and receiving up to hundreds of simultaneous, hetereogeneous sources.

2.2. Support for fine-grained receiver control of sources

Since there may be large numbers of sources, which can be displayed in different layouts, it is imperative that the receiver can easily control which sources are received, and what resolution or quality is desired, for both audio and video. The receiver should also be able to prioritize the source it requests, so that if system limits or bandwidth force a reduction in quality, the sources chosen by the receiver as important will receive the best quality. These details must be exposed to the application via the API.

2.3. Glareless addition and removal of sources

Sources may come and go frequently, as is the case in a conference where various participants are presenting, or an interaction between

multiple distributed conference servers. Because of this, it is desirable that sources can be added to SDP in a way that avoids signaling glare. The m= line approach suffers from this problem, as each m= line is indicated by a numeric index, and either side may add a new m= line as the same numeric index.

2.4. Interworking with legacy devices

When interacting with a legacy application that only knows how to deal with a small number of sources, it must be possible to degrade gracefully to a usable basic experience, where at least a single audio and video source are active on each side, using a typical offer /answer exchange.

2.5. Avoidance of unnecessary port allocation

When there are dozens or hundreds of streams, it is desirable to avoid creating dozens or hundreds of transports, as empirical data shows a clear inverse relationship between number of transports (NAT bindings) and call success rate. While BUNDLE helps avoid creating large numbers of transports, it is also desirable to avoid creating large numbers of ports during call setup, since ICE pacing alone can lead to significant delays, and BUNDLE does not attempt to address this.

2.6. Simple binding of MediaStreamTrack to SDP

In WebRTC, each media source is identified by a MediaStreamTrack object. In order to ensure that the MSTs created by the sender show up at the receiver, each MST's id attribute needs to be reflected in SDP.

2.7. Support for RTX, FEC, simulcast, layered coding

For robust applications, techniques like RTX and FEC are used to protect media, and simulcast/layered coding can be used to provide support to hetereogeneous receivers. It needs to be possible to support these techniques, allow the receipient to optionally use or not use them on a source-by-source basis, and for simulcast/layered scenarios, control which simulcast streams or layers are received.

2.8. Works with or without BUNDLE

While BUNDLE provides a major win in allowing the multiplexing of multiple m= lines over a single transport, it does not solve all transport-related problems. Moreover, like any new technology, it will take time to identify all the edge cases and deployment issues. Therefore not having a hard dependency on BUNDLE, while still keeping the ability to benefit from BUNDLE, is considered advantageous.

3. Detailed Design

3.1. One m= line per [media-type, content] tuple

Regardless of how many sources are offered or used, only one m= line, by default, is used for each media type (e.g. audio, video, or application). If the application wishes, it can request that a given media source be placed onto a separate m= line, by setting a new .content property on the desired MediaStreamTrack; the values for the .content property are those defined for the a=content attribute in [RFC4796]. This helps with certain legacy interop cases, for example, an endpoint who expected a screenshare video source to be signaled on its own m=video line. It is expected that the number of scenarios that require this property setting are fairly limited, primarily to cases like this one. In these situations, the number of m= lines still remains manageable.

By only using a m= line for each media type, as opposed to each media source, this approach reduces the number of transports required to 2 even in complex audio/video cases. Note that in the common case of a call with a single audio and video source, the result is identical to the m= line per source approach, and so there are no issues with legacy devices in this case. In more complex cases, the application can select which of the sources for each m= line should be sent when interacting with a legacy device.

3.2. Transmit sources identified by SSRC in signaling

Media sources are identified in signaling by SSRC, via [RFC5576] a=ssrc attributes. Each provider of MSTs (i.e. endpoint) indicates the MSTs they wish to send in their signaling message, and describes them using one of more SSRCs; when techniques like simulcast or RTX are used, a media source can span multiple SSRCs. Correlation of MediaStreamTracks with SSRCs is performed by the MSID attribute, a per-ssrc attribute defined in [I-D.alvestrand-mmusic-msid] that contains the MST's "id" property, and thereby indicates which MST corresponds to which SSRC. In the example below, the SDP advertises

3 separate audio sources, each identified by a single SSRC.

```
m=audio 49170 RTP/AVP 101       // main audio
a=ssrc:1 msid:left-mic          // declare 3 outgoing audio sources
```

```
a=ssrc:2 msid:center-mic
a=ssrc:3 msid:right-mic
```

Note that because of the signaling of SSRCs, SSRC collision is not a major issue. Each endpoint can ensure that the SSRCs it chooses don't collide, because it already knows about the SSRCs it has already used, as well as the ones the remote side has signaled. In the few cases where it can still occur - perhaps because of simultaneous signaling by both endpoints, upon recognizing the collision, an endpoint can update its send sources to not collide. If a collision occurs on an in-progress media flow, or if an endpoint needs to change SSRCs, for whatever reason, the [RFC5576] a=previous-ssrc attribute can be used to change an existing SSRC to a new value, while retaining continuity of media playout.

3.3. Receive sources requested using remote-ssrc

When the offerer advertises its streams, the remote side can select which of them it wants to receive in its answer message. This is performed via [I-D.lennox-mmusic-sdp-source-selection], which introduces the concept of an a=remote-ssrc attribute. Through use of this attribute, the remote side chooses which of the offered sources should be received, by turning each one on or off, and optionally specifying what resolution, framerate and priority the recipient suggests for a particular source. This can also be used to select simulcast streams, or enabling/disabling of techniques like RTX or FEC. In the example below, the SDP sent by the receiver asks for just one of the offered streams.

```
m=audio 39170 RTP/AVP 101       // main audio
a=remote-ssrc:1 recv:on         // just turn on the center mic
```

The exact rules for how this logic works are specified in the aforementioned draft.

3.4. RTX, FEC, Simulcast

In cases where a media source needs to correspond to more than one RTP flow, e.g. RTX, FEC, or simulcast, the [RFC5576] a=ssrc-group concept is used to create a grouping of SSRCs for a single MST. Each SSRC is declared using a=ssrc attributes, the same MSID is shared between the SSRCs, and the a=ssrc-group attribute defines the behavior of the grouped SSRCs.

These groupings are used to perform demux of the incoming RTP

streams
   and associate them (by SSRC) with their primary flows; this is the

only way that a RTX or FEC stream can be correlated with its primary flow, when many streams are multiplexed in a single RTP session. This multiplexing of RTX and FEC in a single RTP session is already well-defined; RTX SSRC-multiplexing behavior is defined in [RFC4588], and FEC SSRC-multiplexing behavior is defined in [RFC5956].

For multi-resolution simulcast, we can create a similar ssrc-group, and adapt the imageattr attribute defined in [RFC6236] for the a=ssrc line attribute to indicate the send resolution for a given simulcast stream. (This will be added to the source-selection draft). In the example below, the SDP advertises a simulcast of a camera source at two different resolutions, as well as a screenshare source that supports RTX; a=ssrc-group is used to correlate the different SSRCs as part of a single media source.

```
m=video 62537 RTP/SAVPF 96      // main video
a=ssrc:5 msid:center-cam        // one source is simulcasting at two resolutions
a=ssrc:5 imageattr:* [1280, 720]
a=ssrc:51 msid:center-cam       // different SSRC, same MSID for simulcasts
a=ssrc:51 imageattr:* [640, 360]
a=ssrc-group:SIMULCAST 5 51
a=ssrc:8 msid:slides
a=ssrc:9 msid:slides            // RTX SSRC
a=ssrc-group:FID 8 9
```

As specified in [I-D.lennox-mmusic-sdp-source-selection], the usage of RTX and FEC can be enabled or disabled on a per-media-source basis, as the recipient can turn off these secondary flows in its SDP. However, it is not possible to get into undefined situations, such as receiving the FEC flow without the primary flow.

Note that at the wire level, the RTP flows will be the same as BUNDLEd session-multiplexed flows - the flows are all multiplexed in a single RTP session, which means that SSRCs must be used to correlate a RTX or FEC flow with their primary flow. Using payload types for this correlation, while possible in certain simple cases, is not suitable for the general case of many media sources, since individual PTs would have to be declared for every [codec, media source, primary/RTX/FEC] tuple, which quickly becomes unworkable.

4. Signaling Behavior

4.1. Negotiation of new or legacy behavior

In order to know whether a given application supports Plan B, an attribute in the offer is needed. There are various options that could be used for this:

- a=ssrc isn't enough, since you might not have any send streams, and therefore no a=ssrc attributes.
- a=max-*-ssrc could work, but has additional semantics
- a=msid-semantic indicates that you understand MSIDs.

Because understanding MSID is a prerequisite to using plan B, the third option (presence of a=msid-semantic) is recommended.

4.2. New signaling flow

When both sides support Plan B, to properly allow both sides to indicate which MSTs they have, and allow the remote side to select the desired MSTs to receive, a 3-way handshake is needed (this is just math; the offer can't select the answerer's MSTs until they know about them). The expected flow for this would be for the caller to send an offer with its sources, then the callee would send back an answer with the sources it wants the caller to send, followed immediately by an offer with the sources that the callee has available to send. Finally, the answerer will reply back with the sources that it wants to request from the callee. The entire sequence can be done in 1.5 RTT.

Typically a race condition exists between the receipt of the callee answer and the receipt of the callee media. However, this is avoided with Plan B, with its 3-way handshake. In addition, since the sources are known ahead of time by the recipient of said sources, it is prepared to demux them by SSRC without any signaling/media race.

4.3. Legacy signaling flow

In the legacy case, Plan B degrades gracefully back to a single offer-answer sequence. Since there's no brokering of which sources should be sent, the "new" endpoint picks a default media source for each m= line, and upon receiving an answer indicating lack of support for Plan B, it sends just the default sources to the legacy endpoint.
When receiving media from the legacy endpoint, the new endpoint creates a "default" MediaStream (containing a single MediaStreamTrack) for each m= line, just as when talking to any other legacy endpoint, as specified in the MSID draft.

Typically this will result in the negotiation of a single audio and single video MediaStreamTrack. However, if the .content property was used above, additional audio or video MediaStreamTracks could be specifically negotiated for the purpose of communicating with legacy equipment (e.g. an additional screenshare video source).

4.4. Interactions with BUNDLE

Since Plan B already muxes streams of the same media type onto a single m= line, BUNDLE is not as critical to its success as other approaches. Regardless, BUNDLE can be used to get down to just a single transport, by multiplexing the various media types together, and Plan B works well with BUNDLE. Since all streams are pre-identified by their SSRC attributes, their RTP flows can be demuxed easily even when grouped on the same 5-tuple.

5. Full Example

5.1. New endpoint

The example below shows two new endpoints, A and B, establishing a Plan B call with audio, video, screensharing, simulcast, and RTX; B decides to select a subset of the sources that A advertises.

5.1.1. Offer 1 (A->B): (A indicates streams to B)

```
a=msid-semantics:WMS            // I understand SSRCs and MSTs

m=audio 49170 RTP/AVP 101       // main audio
a=ssrc:1 msid:left-mic          // declare 3 outgoing audio sources,
each with unique MSID
a=ssrc:2 msid:center-mic
a=ssrc:3 msid:right-mic
[Candidates]

m=video 62537 RTP/SAVPF 96      // main video
a=ssrc:4 msid:left-cam          // declare 3 outgoing video sources
a=ssrc:5 msid:center-cam        // one source is simulcasting at two
resolutions, same codec
a=ssrc:5 imageattr:* [1280, 720]
a=ssrc:51 msid:center-cam       // different SSRC, same MSID for
simulcasts
a=ssrc:51 imageattr:* [640, 360]
a=ssrc:6 msid:right-cam
a=ssrc-group:SIMULCAST 5 51
[Candidates]

m=video 62538 RTP/SAVPF 96      // presentation
a=content:slides                // [media, content] tuples must be
unique in m= lines
a=ssrc:8 msid:slides            // app decision to do this; could
```

```
have been put in the m=video line above
   a=ssrc:9 msid:slides            // RTX SSRC
```

```
   a=ssrc-group:FID 8 9            // declaration that SSRC 9 is a
repair flow for 8
   [Candidates]
```

5.1.2. Answer 1 (B->A): (B requests streams from A)

```
   a=msid-semantics:WMS            // I understand SSRCs and MSTs

   m=audio 39170 RTP/AVP 101       // main audio
   a=remote-ssrc:1 recv:on         // just turn on the center mic
   [Candidates]

   m=video 52537 RTP/SAVPF 96      // main video
   a=remote-ssrc:5 recv:off        // explicitly turn off the 720p feed
   a=remote-ssrc:51 recv:on        // explicitly turn on the 360p feed
   a=remote-ssrc:51 priority:1     // lower priority than slides (in
this application)
   [Candidates]

   m=video 52538 RTP/SAVPF 96      // presentation
   a=content:slides
   a=remote-ssrc:8 recv:on         // turn on the slides feed with
higher priority
                                   // FID is sent implicitly, unless
explicitly rejected
   a=remote-ssrc:8 priority:2      // (sender uses priority when making
BW decisions)
   [Candidates]
```

5.1.3. Offer 2 (B->A): (B indicates streams to A)

```
   a=msid-semantics:WMS            // I understand SSRCs and MSTs

   m=audio 39170 RTP/AVP 101       // main audio
   a=ssrc:101 msid:center-mic
   a=remote-ssrc:1 recv:on         // just turn on the center mic
   [Candidates]

   m=video 52537 RTP/SAVPF 96      // main video
   a=ssrc:105 msid:center-cam
   a=remote-ssrc:5 recv:off        // explicitly turn off the 720p feed
   a=remote-ssrc:51 recv:on        // explicitly turn on the 360p feed
   a=remote-ssrc:51 priority:1     // lower priority than slides (in
this application)
   [Candidates]

   m=video 52538 RTP/SAVPF 96      // presentation
   a=content:slides
   a=remote-ssrc:8 recv:on         // turn on the slides feed with
higher priority
                                   // FID is sent implicitly (unless
explicitly rejected)
   a=remote-ssrc:8 priority:2      // (sender uses priority when making
BW decisions)
   [Candidates]
```

5.1.4. Answer 2 (A->B): (A requests streams from B)

```
   a=msid-semantics:WMS            // I understand SSRCs and MSTs

   m=audio 49170 RTP/AVP 101       // main audio
   a=ssrc:1 msid:left-mic          // declare 3 outgoing audio sources,
each with unique MSID
   a=ssrc:2 msid:center-mic
   a=ssrc:3 msid:right-mic
   a=remote-ssrc:101 on
   [Candidates]

   m=video 62537 RTP/SAVPF 96      // main video
   a=ssrc:4 msid:left-cam          // declare 3 outgoing video sources
   a=ssrc:5 msid:center-cam        // one source is simulcasting at two
resolutions, same codec
   a=ssrc:5 imageattr:* [1280, 720]
   a=ssrc:51 msid:center-cam       // different SSRC, same MSID for
simulcasts
   a=ssrc:51 imageattr:* [640, 360]
   a=ssrc:6 msid:right-cam
   a=ssrc-group:SIMULCAST 5 51
   a=remote-ssrc:105 on
   [Candidates]

   m=video 62538 RTP/SAVPF 96      // presentation
```

```
   a=content:slides                // [media, content] tuples must be
unique in m= lines
   a=ssrc:8 msid:slides            // app decision to do this; could
have been put in the m=video line above
   a=ssrc:9 msid:slides            // RTX SSRC
```

```
   a=ssrc-group:FID 8 9            // declaration that SSRC 9 is a
repair flow for 8
   [Candidates]
```

5.2. Legacy endpoint

The example below shows a new endpoint, A, and a legacy endpoint, B, establishing a Plan B call with audio, video, and screensharing. Although A supports multiple media sources for audio and video, B receives only A's default streams. However, since screensharing was done as its own m= line, screensharing continues to work with the legacy endpoint.

5.2.1. Offer 1 (A->B): (A indicates streams to B)

```
   a=msid-semantics:WMS            // I understand SSRCs and MSTs

   m=audio 49170 RTP/AVP 101       // main audio
   a=ssrc:1 msid:left-mic          // declare 3 outgoing audio sources,
each
                                   // with unique MSID
   a=ssrc:2 msid:center-mic
   a=ssrc:3 msid:right-mic
   [Candidates]

   m=video 62537 RTP/SAVPF 96      // main video
   a=ssrc:4 msid:left-cam          // declare 3 outgoing video sources
   a=ssrc:5 msid:center-cam        // one source is simulcasting at two
                                   // resolutions, same codec
   a=ssrc:5 imageattr:* [1280, 720]
   a=ssrc:51 msid:center-cam       // different SSRC, same MSID for
simulcasts
   a=ssrc:51 imageattr:* [640, 360]
   a=ssrc:6 msid:right-cam
   a=ssrc-group:SIMULCAST 5 51
   [Candidates]

   m=video 62538 RTP/SAVPF 96      // presentation
   a=content:slides                // [media, content] tuples must be
unique
                                   // in m= lines
   a=ssrc:8 msid:slides            // app decision to do this; could
have been
                                   // put in the m=video line above
   a=ssrc:9 msid:slides            // RTX SSRC
   a=ssrc-group:FID 8 9            // declaration that SSRC 9 is a
repair
                                   // flow for 8
   [Candidates]
```

5.2.2. Answer 1 (B->A): (A accepts B's offer)

```
   // Since endpoint doesn't understand a=ssrc or MSTs, A must decide
   // on a single media source to send for each m=line, e.g. center-
mic,
   // center-cam (360p), slides

   m=audio 39170 RTP/AVP 101       // main audio
   [Candidates]

   m=video 52537 RTP/SAVPF 96     // main video
   [Candidates]

   m=video 52538 RTP/SAVPF 96     // presentation
   a=content:slides
   [Candidates]
```

6. Security Considerations

None.

7. IANA Considerations

None.

8. Acknowledgements

Harald Alvestrand provided review and several suggestions on this document.

9. References

9.1. Normative References

[I-D.alvestrand-mmusic-msid]
Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol", draft-alvestrand-mmusic-msid-01 (work in progress), October 2012.

[I-D.lennox-mmusic-sdp-source-selection]
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-05 (work in progress), October 2012.

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

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

9.2. Informative 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.

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

[RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.

[RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.

Author's Address

Justin Uberti
Google
747 6th St S
Kirkland, WA 98033
USA

Email: justin@uberti.name