Network Working Group C. Jennings
Internet-Draft
Cisco
Intended status: Informational February 25, 2013
Expires: August 29, 2013

Proposed Plan for Usage of SDP and RTP
draft-jennings-rtcweb-plan-01

Abstract

This draft outlines a bunch of the remaining issues in RTCWeb related to how the the W3C APIs map to various usages of RTP and the associated SDP. It proposes one possible solution to that problem and outlines several chunks of work that would need to be put into other drafts or result in new drafts being written. The underlying design guideline is to, as much as possible, re-use what is already defined in existing SDP [RFC4566] and RTP [RFC3550] specifications.

This draft is not intended to become an specification but is meant for working group discussion to help build the specifications. It is being discussed on the rtcweb@ietf.org mailing list though it has topics relating to the CLUE WG, MMUSIC WG, AVT* WG, and WebRTC WG at W3C.

Table of Contents

1. Overview

The reoccurring theme of this draft is that SDP [RFC4566] already has a way of solving many of the problems being discussed at the RTCWeb WG and we SHOULD not try to invent something new but rather re-use the existing methods for describing RTP [RFC3550] media flows.

The general theory is that, roughly speaking, the m-line corresponds to flow of packets that can be handled by the application in the same way. This often results in more m-lines than there are media sources such as microphones or cameras. Forward Error Correction (FEC) is done with multiple M-lines as shown in [RFC4756]. Retransmission (RTX) is done with multiple m-lines as shown in [RFC4588]. Layered coding is done with multiple m-lines as shown in [RFC5583]. Simulcast, which is really just multiple video stream from the same camera, much like layered coding but with no inter m-line dependency, is done with multiple m-lines modeled after the Layered coding defined in in [RFC5583].

The significant addition to SDP semantics is an multi-render media level attribute that allows a device to indicate that it makes sense to simultaneously use multiple stream of video that will be simultaneously displayed but share the same SDP characteristics and semantics such that they can all be negotiated under a single m-line. When using features like RTX, FEC, and Simulcast in a multi-render situation, there needs to be a way to correlate a given related media flow with the correct "base" media-flow. This is accomplished by having the related flows carry, in the CSRC, the SSRC of their base flow. An example SDP might look like as provided in the example Section 7.2.1.

This draft also propose that advanced usages, including WebRTC to WebRTC scenarios, uses a Media Stream Identifier (MSID) that is signaled in SDP and also attempts to negotiate the usage of a RTP header extension to include the MSID in the RTP packet. This resolves many long term issues.

This does results in lots of m lines but all the alternatives designs resulted in an roughly equivalent number of SSRC lines with a possibility of redefining most of the media level attributes. So it's really hard to see the big benefits defining something new over what we have. One of the concerns about this approach is the time to collect all the ICE candidates needed for the initial offer. Section 7.2.1 provides mitigations to reduce the number of ports

needed to be the same as an alternative SSRC based design. This assumes that it is perfectly feasible to transport SDP that much larger than a single MTU. The SIP [RFC3261] usage of SDP has successfully passed over this long ago. In the cases where the SDP

is passed over web mechanisms, it is easy to use compression and the size of SDP is more of an optimization criteria than a limiting issue.

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].

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

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

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.

PC-Stream: Represents stream of data of audio and/or video added to a Peer Connection by local or remote media source(s). A PC-Stream is made up of zero or more PC-Tracks.

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

m-block: An SDP [RFC4566] media description that starts with an m-line and is terminated by either the next m-line or by the end of the session description.

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

Answer: An [RFC3264] SDP message generated by the participant in response to an Offer. An Answer describes participants 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. Requirements

The requirements listed here are a collection of requirements that have come from WebRTC, CLUE, and the general community that uses RTP for interactive communications based on Offer/Answer. It does not try to meet the needs of streaming usages or usages involving multicast. This list also does not try to list every possible requirement but instead outlines the ones that might influence the design.

- Devices with multiple audio and/or video sources

- Devices that display multiple streams of video and/or render multiple streams of audio

- Simulcast, wherein a video from a single camera is sent in a few independent video streams typically at different resolutions and frame rates.

- Layered Codec such as H.264 SVC

- One way media flows and bi-directional media flows

- Support asymmetry, i.e. to send a different number of type of media streams that you receive.

- Mapping W3C PeerConnection (PC) aspects into SDP and RTP. It is important that the SDP be descriptive enough that both sides can get the same view of various identifiers for PC-Tracks, PC-Streams and their relationships.

- Support of Interactive Connectivity Establishment (ICE) [RFC5245]

- Support of multiplexing multiple media flows, possible of different media types, on same 5-tuple.

- Synchronization - It needs to be clear how implementations deal with synchronization, in particular usages of both CNAME and LS group. The sender needs be able to indicate which Media Flows are intended to be synchronized and which are not.

- Redundant codings - The ability to send some media, such as the audio from a microphone, multiple times. For example it may be sent with a high quality wideband codec and a low bandwidth codec. If packets are lost from the high bandwidth steam, the low bandwidth stream can be used to fill in the missing gaps of audio. This is very similar to simulcast.

- Forward Error Correction - Support for various RTP FEC schemes.

- RSVP QoS - Ability to signal various QoS mechanism such Single Reservation Flow (SRF) group

- Desegregated Media (FID group) - There is a growing desire to deal with endpoints that are distributed - for example a video phone where the incoming video is displayed on the an IP TV but the outgoing video comes from a tablet computer. This results in situations where the SDP sets up a session with not all the media transmitted to a single IP address.

- In flight change of codec: Support for system that can negotiate the uses of more than one codec for a given media flow and then the sender can arbitrarily switch between them when they are sending but they only send with one codec as at time.

- Distinguish simulcast (e.g. multiple encoding of same source) from multiple different sources

- Support for Sequential and Parallel forking at the SIP level

- Support for Early Media

- Conferencing environments with Transcoding MCU that decodes/ mixes/ recodes the media

- Conferencing environments with Switching MCU where the MCU mucks the header information of the media and do not decode/recode all the media

4. Background/Solution Overview

The basic unit of media description in SDP is the m-line/m-block. This allows any entity defined by a single m-block to be individually negotiated. This negotiation applies not only to individual sources (e.g., cameras) but also to individual components that come from a single source, such as layers in SVC.

For example, consider negotiation of FEC as defined in [RFC4756].

Offer

```
v=0
o=adam 289083124 289083124 IN IP4 host.example.com
s=ULP FEC Seminar
t=0 0
c=IN IP4 192.0.2.0
a=group:FEC 1 2
a=group:FEC 3 4

m=audio 30000 RTP/AVP 0
a=mid:1

m=audio 30002 RTP/AVP 100
a=rtpmap:100 ulpfec/8000
a=mid:2

m=video 30004 RTP/AVP 31
a=mid:3

m=video 30004 RTP/AVP 101
c=IN IP4 192.0.2.1
a=rtpmap:101 ulpfec/8000
a=mid:4
```

When FEC is expressed this way, the answerer can selectively accept or reject the various streams by setting the port in the m-line to zero. RTX [RFC4588], layered coding [RFC5583], and Simulcast are all handled the same way. Note that while it is also possible to represent FEC and SVC using source-specific attributes [RFC5576], that mechanism is less flexible because it does not permit selective acceptance and rejection as described in [RFC 5576; Section 8]. Most deployed systems which implement FEC, layered coding, etc. do so with each component on a separate m-line.

Unfortunately, this strategy runs into problems when combined with two new features that are desired for WebRTC:

m-line multiplexing (bundle):

The ability to send media described in multiple media over the same 5-tuple.

multi-render:

The ability to have large numbers of multiple similar media flows (e.g., multiple cameras). The paradigmatic case here is multiple video thumbnails.

Obviously, this strategy does not scale to large numbers For instance, consider the case where we want to be able to transmit 35 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.261, H.263, and VP8), then we have just consumed 105 payload types, which exceeds the available dynamic payload space.

In order to resolve this issue, it is necessary to have multiple flows (e.g., multiple thumbnails) indicated by the same m-line and using the same set of payload types (see Section XXX for proposed syntax for this.) Because each source has its own SSRC, it is possible to divide the RTP packets into individual flows. However, this solution still leaves us with two problems:

- How to individually address specific RTP flows in order to, for instance, order them on a page or display flow-specific captions.
- How to determine the relationship between multiple variants of the same stream. For instance, if we have multiple cameras each of which is present in a layered encoding, we need to be able to determine which layers go together.

For reasons described in Section 5, the SSRC learned visa SDP is not suitable for individually addressing RTP flows. Instead, we introduce a new identifier, the MSID, which can be carried both in the SDP and the RTP and therefore can be used to correlate SDP elements to RTP elements. See Section 7.1

By contrast, we can use RTP-only mechanisms to express the correlation between RTP flows: while all the flows associated with a given camera have distinct SSIDs, we can use the CSRC to indicate which flows belong together. This is described in Section 7.2

5. Overall Design

The basic unit of media description in SDP is the m-line/m-block and this document continues with that assumption. In general, different cameras, microphones, etc. are carried on different m-lines. The exceptions to this rule is when using the multi-render extension in which case:

- Multiple sources which are semantically equivalent and multiplexed on a time-wise basis. For instance, if an MCU mixes multiple camera feeds but only some subset is displayed at a time, they can all appear on the same m-line.

By contrast, multiple sources which are semantically distinct cannot appear on the same m-line because that does not allow for clear negotiation of which sources are acceptable, or which sets of RTP SSRCs correspond to which flow.

The second basic assumption is that SSRCs cannot always be safely used to associate RTP flows with information in the SDP. There are two reasons for this. First, in an offer/answer setting, RTP can appear at the offerer before the answer is received; if SSRC information from the offerer is required, then these RTP packets cannot be interpreted. The second reason is that RTP permits SSRCs to be changed at any time.

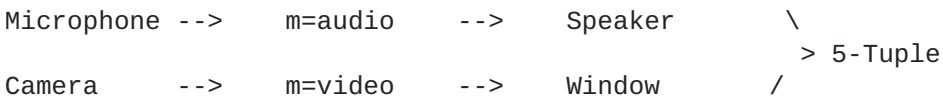
This assumption makes clear why the two exceptions to the "one flow per m-line" rule work. In the case of time-based multiplexing (multi render) of camera sources, all the cameras are equivalent from the receiver's perspective; he merely needs to know which ones to display now and he does that based on which ones have been most recently received. In the case of multiple versions of the same content, payload types or payload types plus SSRC can be used to distinguish the different versions.

6. Example Mappings

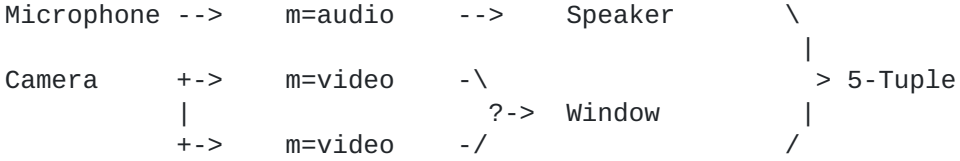
This section shows a number of sample mappings in abstract form.

6.1. One Audio, One Video, No bundle/multiplexing

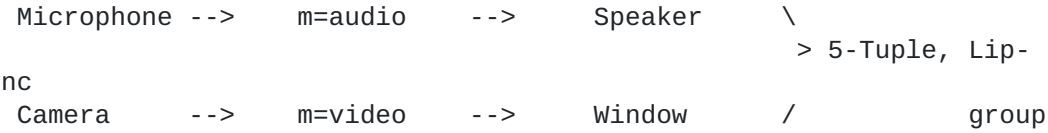
6.2. One Audio, One Video, Bundle/multiplexing



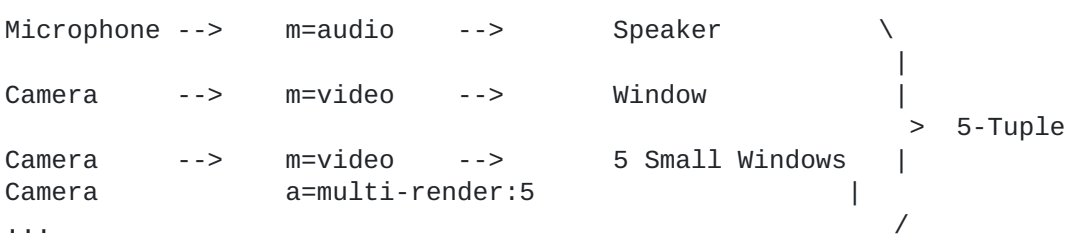
6.3. One Audio, One Video, Simulcast, Bundle/multiplexing



6.4. One Audio, One Video, Bundle/multiplexing, Lip-Sync

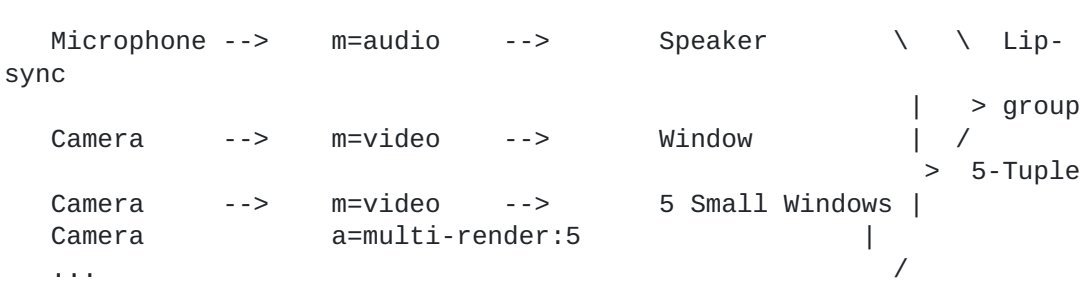


6.5. One Audio, One Active Video, 5 Thumbnails, Bundle/multiplexing



Note that in this case the payload types must be distinct between the two video m-lines, because that is what is used to demultiplex.

6.6. **One Audio, One Active Video,** 5 Thumbnails, Main Speaker Lip-Sync, Bundle/multiplexing



7. Solutions

This section outlines a set of rules for the usage of SDP and RTP that seems to deal with the various problems and issues that have been discussed. Most of these are not new and are pretty much how many systems do it today. Some of them are new, but all the items requiring new standardization work are called out in the Section 9.

Approach:

1. If a system wants to offer to send two sources, such as two camera, it MUST use a separate m-block for each source. The means that each PC-Track corresponds to one or more m-blocks.

2. In cases such as FEC, simulcast, SVC, each repair stream, layer, or simulcast media flow will get an m-block per media flow.

3. If a systems wants to receive two streams of video to display in two different windows or screens, it MUST use separate m-blocks for each unless explicitly signaled to be otherwise (see Section 7.2).

4. Unless explicitly signaled otherwise (see Section 7.2), if a given m-line receives media from multiple SSRCs, only media from the most recently received SSRC SHOULD be rendered and other SSRC SHOULD NOT and if it is video it SHOULD be rendered in the same window or screen.

5. If a camera is sending simulcast video and three resolutions, each resolution MUST get its own m-block and all the three m-blocks will be grouped. A new SDP group will be defined for this.

6. If a camera is using a layered codec with three layers, there MUST be an m-block for each, and they will be grouped using standard SDP for grouping layers.

7. To aid in synchronized playback, there is exactly one, and only one, LS group for each PC-Stream. All the m-blocks for all the PC-Tracks in a given PC-Stream are synchronized so they are all put in one LS group. All the PC-Tracks in a given PC-Stream have the same CNAME. If a PC-Track appears in more than one PC-Stream, then all the PC-Streams with that PC-Track MUST have the same CNAME.

8. One way media MUST use the sendonly or recvonly attributes.

9. Media lines that are not currently in use but may be used later, so that the resources need to be kept allocated, SHOULD use the inactive attribute.

10. If an m-line will not be used, or it is rejected, it MUST have its port set to zero.

11. If a video switching MCU produces a virtual "active speaker" media flow, that media flow should have its own SSRC but include the SSRC of the current speaker's video in the CSRC packets it produces.

12. For each PC-Track, the W3C API MUST provide a way to set and read the CSRC list, set and read the content RFC 4574 "label", and read the SSRC of last packet received on a PC-Track.

13. The W3C API should have a constraint or API method to allow a PC-Stream to indicate the number of multi-render video streams it can accept. Each time a new stream is received up to the maximum, a new PC-Track will be created.

14. Applications MAY signal all the SSRC they intend to send using RFC 5576, but receivers need to be careful in their usage of the SSRC in signaling, as the SSRC can change when there is a collision and it takes time before that will be updated in signaling.

15. Applications can get out of band "roster information" that maps the names of various speakers or other information to the MSID and/or SSRCs that a user is using

16. Applications MAY use RFC 4574 content labels to indicate the purpose of the video. The additional content types, main-left and main-right, need to be added to support two- and three-screen systems.

17. The CLUE WG might want to consider SDP to signal the 3D location and field of view parameters for captures and renderers.

18. The W3C API allows a "label" to be set for the PC-Track. This MUST be mapped to the SDP label attribute.

7.1. Correlation and Multiplexing

The port number that RTP is received on provides the primary mechanism for correlating it to the correct m-line. However, when the port does not uniquely male the RTP packet to the correct m-block

(such as in multiplexing and other cases), the next thing that can be

looked at is the PT number. Finally there are cases where SSRC can be used if that was signaled.

There are some complications when using SSRC for correlation with signaling. First, the offerer may end up receiving RTP packets before receiving the signaling with the SSRC correlation information.
This is because the sender of the RTP chooses the SSRC; there is no way for the receiver to signal how some of the bits in the SSRC should be set. Numerous attempts to provide a way to do this have been made, but they have all been rejected for various reasons, so this situation is unlikely to change. The second issue is that the signaled SSRC can change, particularly in collision cases, and there is no good way to know when SSRC are changing, such that the currently signaled SSRC usage maps to the actual RTP SSRC usage. Finally SSRC does not always provide correlation information between media flows - take for example trying to look at SSRC to tell that an audio media flow and video media flow came from the same camera. The nice thing about SSRC is that they are also included in the RTP.

The proposal here is to extend the MSID draft to meet these needs: each media flow would have a unique MSID and the MSID would have some level of internal structure, which would allow various forms of correlation, including what WebRTC needs to be able to recreate the MS-Stream / MS-Track hierarchy to be the same on both sides. In addition, this work proposes creating an optional RTP header extension that could be used to carry the MSID for a media flow in the RTP packets. This is not absolutely needed for the WebRTC use cases but it helps in the case where media arrives before signaling and it helps resolve a broader category of web conferencing use cases.

The MSID consists of three things and can be extended to have more. It has a device identifier, which corresponds to a unique identifier of the device that created the offer; one or more synchronization context identifiers, which is a number that helps correlate different synchronized media flows; and a media flow identifier. The synchronization identifier and flow identifier are scoped within the context of the device identifier, but the device identifier is globally unique. The suggested device identifier is a 64-bit random number. The synchronization group is an integer that is the same for all media flows that have this device identifier and are meant to be synchronized. Right now there can be more than one synchronization identifier, but the open issues suggest that one would be preferable.
The flow identifier is an integer that uniquely identifies this media

flow within the context of the device identifier.

Open Issues: how to know if the MSID RTP Header Extension should be included in the RTP?

An example MSID for a device identifier of 12345123451234512345, synchronization group of 1, and a media flow id of 3 would be:

a=msid:12345123451234512345 s:1 f:3

When the MSID is used in an answer, the MSID also has the remote device identifier included. In the case where the device ID of the device sending the answer was 22222333334444455555, the MSID would look like:

a=msid:22222333334444455555 s:1 f:3 r:12345123451234512345

Note: The 64 bit size for the device identifier was chosen as it allows less than a one in a million chance of collision with greater than 10,000 flows (actually it allows this probability with more like 6 million flows). Much smaller numbers could be used but 32 bits is probably too small. More discussion on the size of this and the color of the bike shed is needed.

When used in the WebRTC context, each PeerConnection should generate a unique device identifier. Each PC-Stream in the PeerConnection will get a a unique synchronization group identifier, and each PC-Track in the Peer Connection will get a unique flow identifier. Together these will be used to form the MSID. The MSID MUST be included in the SDP offer or answer so that the WebRTC connection on the remote side can form the correct structure of remote PC-Streams and PC-Tracks. If a WebRTC client receives an Offer with no MSID information and no LS group information, it MUST put all the remote PC-Tracks into a single PC-Stream. If there is LS group information but no MSID, a PC-Stream for each LS group MUST be created and the PC-Tracks put in the appropriate PC-Stream.

The W3C specs should be updated to have the ID attribute of the MS-Stream be the MSID with no flow identifier, and the ID attribute of the MS-Track be the MSID.

In addition, the SDP will attempt to negotiate sending the MSID in the RTP using a RTP Header Extension. WebRTC clients SHOULD also include the a=ssrc attributes if they know which SSRC they plan to send but they can not rely on this not changing, being compete, or existing in all offers or answers they receive - particularly when working with SIP endpoints.

When using multiplexing, the SDP MUST be distinct enough where the combination of payload type number and SSRC allows for unique demultiplexing of all the media on the same transport flow without use of MSID though the MSID can help in several use cases.

7.2. Multiple Render

There are cases - such as a grid of security cameras or thumbnails in a video conference - where a receiver is willing to receive and display several media flows of video. The proposal here is to create a new media level attribute called multi-render that includes an integer that indicates how many streams can be rendered at the same time.

As an example of a m-block, a system that could display 16 thumbnails at the same time and was willing to receive H261 or H264 might offer

Offer

```
 m=video 52886 RTP/AVP 98 99
 a=multi-render:16
 a=rtpmap:98 H261/90000
 a=rtpmap:99 H264/90000
 a=fmtp:99 profile-level-id=4de00a;
        packetization-mode=0; mst-mode=NI-T;
        sprop-parameter-sets={sps0},{pps0};
```

When combining this multi-render feature with multiplexing, the answer will might not know all the SSRCs that will be send to this m-block so it is best to use payload type (PT) numbers that are unique for the SDP: the demultiplexing may have to only use the PT if the SSRCs are unknown.

The intention is that the most recently sent SSRC are the ones that are rendered. Some switching MCU will likely only send the correct number of SSRC and not change the SSRC but will instead update the CSRC as the switching MCU select a different participant to include in the particular video stream.

The receiver displays, in different windows, the video from the most recent 16 SSRC to send video to m-block.

This allows a switching MCU to know how many thumbnail type streams would be appropriate to send to this endpoint.

7.2.1. Complex Multi Render Example

The following shows a single multi render m-line that can display up to three video streams, and send 3 streams, and support 2 layers of simulcast with FEC on the high resolution layer and bundle. Note that only host candidates are provided for the FEC and lower resolution simulcast so if the device is behind a NAT, those streams will not be used.

Offer

```
v=0
o=alice 20519 0 IN IP4 0.0.0.0
s=ULP FEC
t=0 0
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
c= IN IP4 24.23.204.141
a=group:BUNDLE vid1 vid2 vid3
a=group:FEC vid1 vid2
a=group:SIMULCAST vid1 vid3

m=video 62537 RTP/SAVPF 96
a=mid:vid1
a=multi-render:3
a=rtcp-mux
a=msid:12345123451234512345 s:1 f:1
a=rtpmap:96 VP8/90000
a=fmtp:96 max-fr=30;max-fs=3600;
a=imageattr:96 [x=1280,y=720]
a=candidate:0 1 UDP  2113667327 192.168.1.4 62537 typ host
a=candidate:1 1 UDP  694302207 24.23.204.141 62537
                     typ srflx raddr 192.168.1.4 rport 62537
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host
a=candidate:1 2 UDP  1694302206 24.23.204.141 64678
                     typ rflx raddr 192.168.1.4 rport 64678

m=video 62541 RTP/SAVPF 97
a=mid:vid2
a=multi-render:3
a=rtcp-mux
a=msid:34567345673456734567 s:1 f:2
a=rtpmap:97 uplfec/90000
a=candidate:0 1 UDP 2113667327 192.168.1.4 62541 typ host

m=video 62545 RTP/SAVPF 98
a=mid:vid3
a=multi-render:3
a=rtcp-mux
a=msid:333444558899000991122 s:1 f:3
a=rtpmap:98 VP8/90000
a=fmtp:98 max-fr=15;max-fs=300;
a=imageattr:96 [x=320,y=240]
a=candidate:0 1 UDP 2113667327 192.168.1.4 62545 typ host
```

The following shows an answer to the above offer that accepts everything and plans to send video from five different cameras in to this m-line (but only three at a time).

Answer

```
v=0
o=Bob 20519 0 IN IP4 0.0.0.0
s=ULP FEC
t=0 0
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:
                    c9:c7:70:9d:1f:66:79:a8:07
c= IN IP4 98.248.92.77
a=group:BUNDLE vid1 vid2 vid3
a=group:FEC vid1 vid2
a=group:SIMULCAST vid1 vid3

m=video 42537 RTP/SAVPF 96
a=mid:vid1
a=multi-render:3
a=rtcp-mux
a=msid:54321543215432154321 s:1 f:1 r:12345123451234512345
a=rtpmap:96 VP8/90000
a=fmtp:96 max-fr=30;max-fs=3600;
a=imageattr:96 [x=1280,y=720]
a=candidate:0 1 UDP 2113667327 192.168.1.7 42537 typ host
a=candidate:1 1 UDP 1694302207 98.248.92.77 42537
                    typ srflx raddr 192.168.1.7 rport 42537
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065
                    typ srflx raddr 192.168.1.7 rport 60065

m=video 42539 RTP/SAVPF 97
a=mid:vid2
a=multi-render:3
a=rtcp-mux
a=msid:11111122222233333444444 s:1 f:2 r:34567345673456734567
a=rtpmap:97 uplfec/90000
a=candidate:0 1 UDP 2113667327 192.168.1.7 42539 typ host

m=video 42537 RTP/SAVPF 98
a=mid:vid3
a=multi-render:3
a=rtcp-mux
a=msid:777777888888999999111111 s:1 f:3 r:333444558899000991122
a=rtpmap:98 VP8/90000
```

```
a=fmtp:98 max-fr=15;max-fs=300;
a=imageattr:98 [x=320,y=240]
a=candidate:0 1 UDP 2113667327 192.168.1.7 42537 typ host
a=candidate:1 1 UDP 1694302207 98.248.92.77 42537
                   typ srflx raddr 192.168.1.7 rport 42537
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065
                   typ srflx raddr 192.168.1.7 rport 60065
```

The following shows an answer to the above by a client that does not support simulcast, FEC, bundle, or msid.

Answer

```
v=0
o=Bob 20519 0 IN IP4 0.0.0.0
s=ULP FEC
t=0 0
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:
                   c9:c7:70:9d:1f:66:79:a8:07
c= IN IP4 98.248.92.77

m=video 42537 RTP/SAVPF 96
a=mid:vid1
a=rtcp-mux
a=recvonly
a=rtpmap:96 VP8/90000
a=fmtp:96 max-fr=30;max-fs=3600;
a=candidate:0 1 UDP 2113667327 192.168.1.7 42537 typ host
a=candidate:1 1 UDP 1694302207 98.248.92.77 42537
                   typ srflx raddr 192.168.1.7 rport 42537
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065
                   typ srflx raddr 192.168.1.7 rport 60065

m=video 0 RTP/SAVPF 97
a=mid:vid2
a=rtcp-mux
a=rtpmap:97 uplfec/90000

m=video 0 RTP/SAVPF 98
a=mid:vid3
a=rtcp-mux
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=428014;
```

max-fs=3600; max-mbps=108000; max-br=14000

7.3. Dirty Little Secrets

If SDP offer/answers are of type AVP or AVPF but contain a crypto of fingerprint attribute, they should be treated as if they were SAVP or SAVPF respectively. The Answer should have the same type as the offer but for all practical purposes the implementation should treat it as the secure variant.

If SDP offer/answers are of type AVP or SAVP, but contain an a=rtcp-fb attribute, they should be treated as if they were AVPF or SAVPF respectively. The SDP Answer should have the same type as the Offer but for all practical purposes the implementation should treat it as the feedback variant.

If an SDP Offer has both a fingerprint and a crypto attribute, it means the Offerer supports both DTLS-SRTP and SDES and the answer should select one and return an Answer with only an attribute for the selected keying mechanism.

These may not look appealing but the alternative is to make cap-neg mandatory to implement in WebRTC.

7.4. Open Issues

What do do with unrecognized media received at W3C PerrConnection level? Suggestion is it creates a new track in whatever stream the MSID would indicate if present and the default stream if no MSID header extension in the RTP.

7.5. Confusions

You can decrypt DTLS-SRTP media before receiving an answer, you can't determine if it is secure or not till you have the fingerprint and have verified it

You can use RTCP-FB to do things like PLI without signaling the SSRC. The PLI packets gets the sender SSRC from the incoming media that is trying to signal the PLI for.

8. Examples

Example of a video client joining a video conference. The client can produce and receive two streams of video, one from the slides and the other of the person. The video of the person is synchronized with the audio. In addition, the client can display up to 10 thumbnails of video. The main video is simulcast at HD size and a thumbnail size.

Offer

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
a=group:LS 1,2,3
a=group:SIMULCAST 2,3

m=audio 49170 RTP/AVP 96      <- This is the Audio
a=mid:1
a=rtpmap:96 iLBC/8000
a=content:main

m=video 51372 RTP/AVP 97      <- This is the main video
a=mid:2
a=rtpmap:97 VP8/90000
a=fmtp:97 max-fr=30;max-fs=3600;
a=imageattr:97 [x=1080,y=720]
a=content:main

m=video 51372 RTP/AVP 98       <- This is the slides
a=mid:2
a=rtpmap:98 VP8/90000
a=fmtp:98 max-fr=30;max-fs=3600;
a=imageattr:98 [x=1080,y=720]
a=content:slides

m=video 51372 RTP/AVP 99       <- This is the simulcast of main
a=mid:3
a=rtpmap:99 VP8/90000
a=fmtp:99 max-fr=15;max-fs=300;
a=imageattr:99 [x=320,y=240]

m=video 51372 RTP/AVP 100      <- This is the 10 thumbnails
a=mid:4
a=multi-render:10
a=recvonly
a=rtpmap:100 VP8/90000
a=fmtp:100 max-fr=15;max-fs=300;
a=imageattr:100 [x=320,y=240]
```

Example of a three-screen video endpoint connecting to a two-screen system which ends up selecting the left and middle screens.

Offer

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
a=rtcp-fb

m=audio 49100 RTP/SAVPF 96
a=rtpmap:96 iLBC/8000

m=video 49102 RTP/SAVPF 97
a=content:main
a=rtpmap:97 H261/90000

m=video 49104 RTP/SAVPF 98
a=content:left
a=rtpmap:98 H261/90000

m=video 49106 RTP/SAVPF 99
a=content:right
a=rtpmap:99 H261/90000
```

Answer

```
v=0
o=bob 2808844564 2808844565 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t= 0 0
a=rtcp-fb

m=audio 50100 RTP/SAVPF 96
a=rtpmap:96 iLBC/8000

m=video 50102 RTP/SAVPF 97
a=content:main
a=rtpmap:97 H261/90000

m=video 50104 RTP/SAVPF 98
a=content:left
a=rtpmap:98 H261/90000

m=video 0 RTP/SAVPF 99
a=content:right
a=rtpmap:99 H261/90000
```

Example of a client that supports SRTP-DTLS and SDES connecting to a client that supports SRTP-DTLS.

Offer

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0

m=audio 49170 RTP/AVP 99
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=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=rtpmap:99 iLBC/8000

m=video 51372 RTP/AVP 96
a=fingerprint:sha-1 92:81:49:83:4a:23:0a:0f:1f:9d:f7:
                     c0:c7:70:9d:1f:66:79:a8:07
a=crypto:1 AES_CM_128_HMAC_SHA1_32
  inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=rtpmap:96 H261/90000
```

Answer

```
v=0
o=bob 2808844564 2808844565 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0

m=audio 49172 RTP/AVP 99
a=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=rtpmap:99 iLBC/8000

m=video 51374 RTP/AVP 96
a=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=rtpmap:96 H261/90000
```

9. Tasks

This section outlines work that needs to be done in various specifications to make the proposal here actually happen.

Tasks:

1. Extend the W3C API to be able to set and read the CSRC list for a PC-Track.

2. Extend the W3C API to be able to read SSRC of last RTP packed received.

3. Write an RTP Header Extension draft to cary the MSID.

4. Fix up MSID draft to align with this proposal.

5. Write a draft to add left, right to the SDP content attribute. Add the stuff to the W3C API to read and write this on a track.

6. Write a draft on SDP "SIMULCAST" group to signal multiple m-block as are simulcast of same video content.

7. Complete the bundle draft.

8. Provide guidance for ways to use SDP for reduced glare when adding of one way media streams.

9. Write a draft defining the multi render attribute.

10. Change W3C API to say that a PC-Track can be in only one PeerConnection or make an object inside the PeerConnection for each track in the PC that can be used to set constraints and settings and get information related to the RTP flow.

11. Sort out how to tell a PC-Track, particularly one meant for receiving information, that it can do simulcast, layered coding, RTX, FEC, etc.

10. Security Considerations

TBD

11. IANA Considerations

This document requires no actions from IANA.

12. Acknowledgments

I would like to thank Suhas Nandakumar, Eric Rescorla, Charles Eckel,
Mo Zanaty, and Lyndsay Campbell for help with this draft.

13. Open Issues

The overall solution is complicated considerably by the fact that WebRTC allows a PC-Track to be used in more than one PC-Stream but requires only one copy of the RTP data for the track to be sent. I am not aware of any use case for this and think it should be removed.

If a PC-Track needs to be synchronized with two different things, they should all go in one PC-Stream instead of two.

14. Existing SDP

The following shows some examples of SDP today that any new system needs to be able to receive and work with in a backwards compatible way.

14.1. Multiple Encodings

Multiple codecs accepted on same m-line [RFC4566].

Offer

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0

m=audio 49170 RTP/AVP 99
a=rtpmap:99 iLBC/8000

m=video 51372 RTP/AVP 31 32
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

Answer

```
v=0
o=bob 2808844564 2808844565 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0

m=audio 49172 RTP/AVP 99
a=rtpmap:99 iLBC/8000

m=video 51374 RTP/AVP 31 32
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

This means that a sender can switch back and forth between H261 and MVP without any further signaling. The receiver MUST be capable of receiving both formats. At any point in time, only one video format is sent, thus implying that only one video is meant to be displayed.

14.2. Forward Error Correction

Multiple m-blocks identified with respective "mid" grouped to indicate FEC operation using FEC-FR semantics defined in [RFC5956].

Offer

```
v=0
o=ali 1122334455 1122334466 IN IP4 fec.example.com
s=Raptor RTP FEC Example
t=0 0
a=group:FEC-FR S1 R1

m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=fec-source-flow: id=0
a=mid:S1

m=application 30000 RTP/AVP 110
c=IN IP4 233.252.0.2/127
a=rtpmap:110 raptorfec/90000
a=fmtp:110 raptor-scheme-id=1; Kmax=8192; T=128;
P=A; repair-window=200000
a=mid:R1
```

14.3. Same Video Codec With Different Settings

This example shows a single codec,say H.264, signaled with different settings [RFC4566].

Offer

```
v=0

m=video 49170 RTP/AVP 100 99 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E; packetization-mode=0;
sprop-parameter-sets=Z0IACpZTBYmI,aMljiA==
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=42A01E; packetization-mode=1;
sprop-parameter-sets=Z0IACpZTBYmI,aMljiA==
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42A01E; packetization-mode=2;
sprop-parameter-sets=Z0IACpZTBYmI,aMljiA==;
sprop-interleaving-depth=45; sprop-deint-buf-req=64000;
sprop-init-buf-time=102478; deint-buf-cap=128000
```

14.4. Different Video Codecs With Different Resolutions Formats

The SDP below shows some m-blocks with various ways to specify resolutions for video codecs signaled [RFC4566].

Offer

```
m=video 49170 RTP/AVP 31
a=rtpmap:31 H261/90000
a=fmtp:31 CIF=2;QCIF=1;D=1

m=video 49172 RTP/AVP 99
a=rtpmap:99 jpeg2000/90000
a=fmtp:99 sampling=YCbCr-4:2:0;width=128;height=128

m=video 49174 RTP/AVP 96
a=rtpmap:96 VP8/90000
a=fmtp:96 max-fr=30;max-fs=3600;
a=imageattr:96 [x=1280,y=720]
```

14.5. Lip Sync Group

[RFC5888] grouping semantics for Lip Synchronization between audio and video

Offer

```
v=0
o=Laura 289083124 289083124 IN IP4 one.example.com
c=IN IP4 192.0.2.1
t=0 0
a=group:LS 1 2

m=audio 30000 RTP/AVP 0
a=mid:1

m=video 30002 RTP/AVP 31
a=mid:2
```

14.6. BFCP

[RFC4583] defines SDP format for Binary Floor Control Protocol (BFCP)
as shown below

Offer

```
m=application 50000 TCP/TLS/BFCP *
a=setup:passive
a=connection:new
a=fingerprint:SHA-1 \
4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=floorctrl:s-only
a=confid:4321
a=userid:1234
a=floorid:1 m-stream:10
a=floorid:2 m-stream:11

m=audio 50002 RTP/AVP 0
a=label:10

m=video 50004 RTP/AVP 31
a=label:11
```

Answer

```
m=application 50000 TCP/TLS/BFCP *
a=setup:passive
a=connection:new
a=fingerprint:SHA-1 \
4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=floorctrl:s-only
a=confid:4321
a=userid:1234
a=floorid:1 m-stream:10
a=floorid:2 m-stream:11

m=audio 50002 RTP/AVP 0
a=label:10

m=video 50004 RTP/AVP 31
a=label:11
```

14.7. Retransmission

The SDP given below shows SDP signaling for retransmission of the original media stream(s) as defined in [RFC4756]

Offer

```
v=0
o=mascha 2980675221 2980675778 IN IP4 host.example.net
c=IN IP4 192.0.2.0
a=group:FID 1 2
a=group:FID 3 4

m=audio 49170 RTP/AVPF 96
a=rtpmap:96 AMR/8000
a=fmtp:96 octet-align=1
a=rtcp-fb:96 nack
a=mid:1

m=audio 49172 RTP/AVPF 97
a=rtpmap:97 rtx/8000
a=fmtp:97 apt=96;rtx-time=3000
a=mid:2

m=video 49174 RTP/AVPF 98
a=rtpmap:98 MP4V-ES/90000
a=rtcp-fb:98 nack
a=fmtp:98 profile-level-id=8;config=0101000001200088400668 2C209\
   0A21F
a=mid:3

m=video 49176 RTP/AVPF 99
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=98;rtx-time=3000
a=mid:4
```

Note that RTX RFC also has the following SSRC multiplexing example but this is meant for declarative use of SDP as there was no way in this RFC to accept, reject, or otherwise negotiate this in a an offer / answer SDP usage.

SDP

```
v=0
o=mascha 2980675221 2980675778 IN IP4 host.example.net
c=IN IP4 192.0.2.0

m=video 49170 RTP/AVPF 96 97
a=rtpmap:96 MP4V-ES/90000
a=rtcp-fb:96 nack
a=fmtp:96 profile-level-id=8;config=01010000012000884006682C209\
  0A21F
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96;rtx-time=3000
```

14.8. Layered coding dependency

[RFC5583] "depend" attribute is shown here to indicate dependency between layers represented by the individual m-blocks

Offer

```
a=group:DDP L1 L2 L3

m=video 20000 RTP/AVP 96 97 98
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4de00a; packetization-mode=0;
  mst-mode=NI-T; sprop-parameter-sets={sps0},{pps0};
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=4de00a; packetization-mode=1;
  mst-mode=NI-TC; sprop-parameter-sets={sps0},{pps0};
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=4de00a; packetization-mode=2;
  mst-mode=I-C; init-buf-time=156320;
  sprop-parameter-sets={sps0},{pps0};
a=mid:L1

m=video 20002 RTP/AVP 99 100
a=rtpmap:99 H264-SVC/90000
a=fmtp:99 profile-level-id=53000c; packetization-mode=1;
  mst-mode=NI-T; sprop-parameter-sets={sps1},{pps1};
a=rtpmap:100 H264-SVC/90000
a=fmtp:100 profile-level-id=53000c; packetization-mode=2;
  mst-mode=I-C; sprop-parameter-sets={sps1},{pps1};
a=mid:L2
a=depend:99 lay L1:96,97; 100 lay L1:98

m=video 20004 RTP/AVP 101
a=rtpmap:101 H264-SVC/90000
a=fmtp:101 profile-level-id=53001F; packetization-mode=1;
  mst-mode=NI-T; sprop-parameter-sets={sps2},{pps2};
a=mid:L3
a=depend:101 lay L1:96,97 L2:99
```

14.9. SSRC Signaling

[RFC5576] "ssrc" attribute is shown here to signal synchronization sources in a given RTP Session

Offer

```
m=video 49170 RTP/AVP 96
a=rtpmap:96 H264/90000
a=ssrc:12345 cname:user@example.com
a=ssrc:67890 cname:user@example.com
```

This indicates what the sender will send. It's at best a guess because in the case of SSRC collision, it's all wrong. It does not

allow one to reject a stream. It does not mean that both streams are displayed at the same time.

14.10. Content Signaling

[RFC4796] "content" attribute is used to specify the semantics of content represented by the video streams.

Offer

```
v=0
o=Alice 292742730 29277831 IN IP4 131.163.72.4
s=Second lecture from information technology
c=IN IP4 131.164.74.2
t=0 0

m=video 52886 RTP/AVP 31
a=rtpmap:31 H261/9000
a=content:slides

m=video 53334 RTP/AVP 31
a=rtpmap:31 H261/9000
a=content:speaker

m=video 54132 RTP/AVP 31
a=rtpmap:31 H261/9000
a=content:sl
```

15. References

15.1. 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.

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

15.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, 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.

[RFC4583] Camarillo, G., "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", RFC 4583, November 2006.

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

[RFC4756] Li, A., "Forward Error Correction Grouping Semantics in Session Description Protocol", RFC 4756, November 2006.

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

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

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

[RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.

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

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

Author's Address

Cullen Jennings
Cisco
400 3rd Avenue SW, Suite 350
Calgary, AB T2P 4H2
Canada

Email: fluffy@iii.ca