

MMUSIC WG
Internet-Draft
Intended status: Informational
Expires: December 30, 2013

R. Even
Huawei Technologies
J. Lennox
Vidyo
Q. Wu
Huawei Technologies
June 28, 2013

The Session Description Protocol (SDP) Application Token Attribute
draft-even-mmusic-application-token-00.txt

Abstract

The RTP fixed header includes the payload type number and the SSRC values of the RTP stream. RTP defines how you de-multiplex streams within an RTP session, but in some use cases applications need further identifiers in order to identify the application semantics associated with particular streams within the session.

This document defines a mechanism to provide the mapping between the SSRCs of RTP streams and the application semantics by defining extensions to RTP and RTCP messages.

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 December 30, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Proposal for an Application ID token	4
3.1. RTCP SDES message	6
3.2. RTP Header Extension	6
4. Using Application ID token	7
5. Acknowledgements	8
6. IANA Considerations	9
7. Security Considerations	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	10

1. Introduction

The RTP [RFC3550] header includes the payload type number and the SSRC values of the RTP stream. RTP defines how you de-multiplex streams within an RTP session, but in some use cases, applications need further identifiers in order to identify semantics associated with particular streams within the session.

There is ongoing work to define how to support, using SDP [RFC4566], multiple RTP media streams in one or more m-lines that define a single RTP session (as specified in [RFC3550]). The work is addressing the WebRTC architecture [I-D.ietf-rtcweb-overview], and some work will be needed when looking for a general solution in MMUSIC that can be used for non-WebRTC systems.

RTCWEB Plan A [I-D.roach-rtcweb-plan-a] that an m-line in SDP represents a single RTP stream. De-multiplexing is done by payload type (PT) number (which MUST be unique), and if unique PTs are not feasible, use SSRC information in the SDP to identify the RTP stream.

RTCWEB Plan B [I-D.uberti-rtcweb-plan] takes a different approach, and creates a hierarchy within SDP; an 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.

Each m-line defines multiple RTP streams. This requires that the SSRCs of all RTP streams in the session are declared before they appear as RTP streams.

No plan [I-D.ivov-rtcweb-noplan] proposes using a single m-line for each media type but does not require that all SSRCs will be declared in the SDP. The de-multiplexing is done based on the unique PT numbers and the mapping of SSRC to the application usage may be done by application protocol. In web application, for example, the application specific signaling may use something like { "leftSSRC": "1234", "rightSSRC": "5678" }.

Some applications may require more information about the usage of the RTP streams. For example, RTP streams from different cameras that need to be identified by the application in order to render them correctly, or a source that can send multiple versions of the same stream in different resolutions (Simulcast [I-D.westerlund-avtcore-rtp-simulcast]).

SDP provides in [RFC4574] a "label" attribute that contains a token defined by an application and is used in its context. "Label" can be attached to m-lines in multiple SDP documents allowing the application to logically identify the media streams across SDP sessions when necessary. The "label" attribute is a token and does not provide any information about the content of the stream. [RFC4796] defines the "content" attribute providing information about the content of the stream, currently there is a small set of values for the content attribute.

Both "label" and "content" attribute are SDP media-level attributes, so when an SDP m-line supports multiple RTP streams, this value is applicable to all sources described by the SDP m-line.

There is a need to have a token that will allow the mapping between a single source (identified by an SSRC) in an m-line to the application logic (Source may be a single RTP stream identified by a unique SSRC). For example, SSRC1 is the RTP stream from the left camera and SSRC2 is the RTP stream from the right camera both can be specified in a single SDP m-line and may have the same PT number.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119[RFC2119] and indicate requirement levels for compliant RTP implementations.

3. Proposal for an Application ID token

As we saw in the previous section, there are tokens defined that could be used for the mapping, but they have existing usages and semantics, and tend to apply at media-level rather than source-level. In order to avoid overload of existing attributes, it is better to have a new token attribute that can identify a specific source corresponding to the application. This document defines such a new token, called "AppID".

AppID is a general-purpose token associated with an RTP stream, allowing the semantics of the stream with a token to be defined by the application. This token may be mapped, for example, to a CLUE media capture using CLUE protocol [I-D.ietf-clue-framework], or to a specific resolution in a simulcast application described in the SDP .

The token is chosen by the sender, and represents the RTP stream that will be sent to the receiver.

The proposed token can be sent using SDP, RTCP SDES messages [RFC3550], or an RTP header extension [RFC5285]

The SSRC mapping may be available to the receiver when receiving the RTP stream through the RTP header extension, but may also be available ahead of time via an RTCP SDES message conveyed before the source started sending, even if the receiver has not seen any RTP packets from this source like in a multipoint conference or in the SDP description.

The receiver can receive new sources that may be of two kinds.

- o A new RTP stream replacing an existing RTP stream, in which case the AppID of the replaced RTP stream will be assigned to the new SSRC.
- o A new RTP stream requiring a different AppID, for example, when adding a presentation stream to an existing call with two video cameras from a room.

The solution should support a RTP session as described using SDP. The RTP session may be specified using a single SDP m-line, or using Bundle [I-D.ietf-mmusic-sdp-bundle-negotiation], using more than one m-line. In the latter case, if the SSRCs of all RTP streams are not known in advance, the AppIDs associated with each m-line need to be available to the receiver in order to map each SSRC to a specific m-line configuration.

To support these cases the document defines a new SDP media level attribute `a=appID` that can be used to list all the appIDs that an application may use.

The appID syntax provides a token identifier and optional SDP attributes that describe the application usage if exists in SDP. Application usage in SDP may be, for example, an image attribute describing a simulcast application usage [`I-D.westerlund-avtcore-rtp-simulcast`].

Each value of the AppID maps to one SSRC at a time. When a new SSRC is mapped to an existing AppID using an RTP header extension or SDES message, it replaces the previous RTP stream for this application usage.

The formal representation of the appID token is:

```
appid-attribute = "appID:" tokenlist [SP attribute]

tokenlist = token *("," token)

; The base definition of "attribute" is in [RFC4566].

; (It is the content of "a=" lines.)
```

Examples:

The SSRCs of the streams are not known when the SDP offer is sent, two appID are specified and can be used for mapping to specific SSRCs in the application.

```
m=video 49200 RTP/AVP 99

a=rtpmap:99 H264/90000

a=appID:2,3
```

The second example is when the application usage of the RTP steam is specified using SDP to provide different image resolutions.

```
m=video 49200 RTP/AVP 98, 99

a=rtpmap:98 H264/90000

a=rtpmap:99 H264/90000

a=appID:2 imageattr:98 send [x=480,y=320] recv *
```

```
a=appID:3 imageattr:99 send [x=800,y=640] recv *
```

3.1. RTCP SDES message

The document specify a new RTCP SDES message

```

0                               1                               2                               3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   AppID = XXX   |   length   |AppID token|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   ....   |

```

This AppID is the same token as defined in the new SDP attribute and will also be used in the RTP header extension.

This SDES message MAY be sent in a compound RTCP packet based on the application need.

3.2. RTP Header Extension

The Application ID could be carried within the RTP header extension field, using [RFC5285] two bytes header extension.

This is negotiated within the SDP i.e.

```
a=extmap:1 urn:ietf:params:rtp-hdrext:App-ID
```

Packets tagged by the sender with the AppID will then contain a header extension as shown below

```

0                               1                               2                               3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ID=1 |   Len=1   |   AppID   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| AppID ..... |
+---+---+---+---+---+

```

To add or modify the AppID by an intermediary can be an expensive operation, particularly if SRTP is used to authenticate the packet. Modification to the contents of the RTP header requires a re-authentication of the complete packet, and this could prove to be a limiting factor in the throughput of a multipoint device.

There is no need to send the AppID header extension with all RTP packets. Senders MAY choose to send it only when a new SSRC is sent, or when an SSRC changes its association to an AppID. If such a mode is being used, the header extension SHOULD be sent in the first few RTP packets to reduce the risk of losing it due to packet loss. For codecs with decoder refresh points (such as I-Frames in video codecs), senders also SHOULD send the AppID header extension along with the packets carrying the decoder refresh.

4. Using Application ID token

The usage of mapping may depend on the de-multiplexing of the RTP streams in the SDP m-lines. Currently we have three options discussed based on input from the RTCweb WG.

For plan A [I-D.roach-rtcweb-plan-a], since each RTP stream is described by a specific m-line it will be enough to have a media level token for mapping the sent stream.

Only need for example:

```
m=video 49200 RTP/AVP 99
```

```
a=rtpmap:99 H264/90000
```

```
a=appID 2
```

For plan B [I-D.uberti-rtcweb-plan] which adds another level of RTP stream description, the mapping of SSRC to the application will need to be at the SSRC level base on [RFC5576] since all SSRCs are specified in the m-line. The document addresses the mapping of SSRCs using the SSRC attribute but uses the msid [I-D.ietf-mmusic-msid] that defines a specific semantics for each SSRC. The following offer example is using RFC5576 to provide source specific attribute identifier.

```
m=video 49200 RTP/AVP
```

```
a=rtpmap:99 H264/90000
```

```
a=max-send-ssrc:{*:3}
```

```
a=max-recv-ssrc:{*:3}
```

```
a=ssrc:11111 AppID:1
```

```
a=ssrc:22222 AppID:2
```

```
a=ssrc:33333 AppID:3
```

When using noplan [I-D.ivov-rtcweb-noplan] in MMUSIC, not all SSRCs will be known ahead of time. For example, in the following SDP the offer offers either two streams with the same resolution (for example two cameras) or two streams with different resolutions.

```
m=video 5002 RTP/SAVPF 98
```

```
a=rtpmap:98 H264/90000
```

```
a= appID 1,2 imageattr:98 send [x=800,y=640,sar=1.1,q=0.6] recv *
```

```
a= appID 3,4 imageattr:98 send [x=480,y=320] recv *
```

```
a=max-send-ssrc:{*:2}
```

In the CLUE WG case the mapping is from an RTP stream to a CLUE media capture specified in the CLUE framework [I-D.ietf-clue-framework]. The SSRCs of all streams may be known like in PLAN B but there are cases where the SDP may not be available so a pre-announce is recommended like in the following example.

```
m=video 49200 RTP/AVP
```

```
a=rtpmap:99 H264/90000
```

```
a=max-send-ssrc:{*:5}
```

```
a=max-recv-ssrc:{*:3}
```

```
a=ssrc:11111 AppID:1
```

```
a=ssrc:22222 AppID:2
```

```
a=ssrc:33333 AppID:3
```

```
a=appID 4, 5
```

The pre-announce is needed since the new RTCP SDES message includes only the SSRC and the appID but not the PT. A receiver of the SDES message will be able to map the SSRC to a codec configuration based on the SDP pre-announced tokens.

5. Acknowledgements

Place Holder

6. IANA Considerations

TBD

7. Security Considerations

TBD.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

8.2. Informative References

- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-10 (work in progress), May 2013.
- [I-D.ietf-mmusic-msid]
Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol", draft-ietf-mmusic-msid-00 (work in progress), February 2013.
- [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-04 (work in progress), June 2013.
- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-06 (work in progress), February 2013.
- [I-D.iovov-rtcweb-noplan]
Ivov, E., Marocco, E., and P. Thatcher, "No Plan: Economical Use of the Offer/Answer Model in WebRTC"

Sessions with Multiple Media Sources", draft-ivov-rtcweb-noplan-01 (work in progress), June 2013.

[I-D.roach-rtcweb-plan-a]

Roach, A. and M. Thomson, "Using SDP with Large Numbers of Media Flows", draft-roach-rtcweb-plan-a-00 (work in progress), May 2013.

[I-D.uberti-rtcweb-plan]

Uberti, J., "Plan B: a proposal for signaling multiple media sources in WebRTC.", draft-uberti-rtcweb-plan-00 (work in progress), May 2013.

[I-D.westerlund-avtcore-rtp-simulcast]

Westerlund, M., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP Sessions", draft-westerlund-avtcore-rtp-simulcast-02 (work in progress), February 2013.

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

[RFC4574] Levin, O. and G. Camarillo, "The Session Description Protocol (SDP) Label Attribute", RFC 4574, August 2006.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

[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.

Authors' Addresses

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Qin Wu
Huawei Technologies

Email: bill.wu@huawei.com

MMUSIC Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2014

R. Gellens
Qualcomm Technologies Inc.
July 14, 2013

Negotiating Human Language Using SDP
draft-gellens-mmusic-negotiating-human-language-01

Abstract

Users have various human (natural) language needs, abilities, and preferences regarding spoken, written, and signed languages. When establishing interactive communication "calls" there needs to be a way to communicate and ideally match (i.e., negotiate) the caller's language preferences with the capabilities of the called party. This is especially important with emergency calls, where a call can be routed to a Public Safety Answering Point (PSAP) or call taker capable of communicating with the user, or a translator or relay operator can be bridged into the call during setup, but this applies to non-emergency calls as well (as an example, when calling a company call center).

This document describes the need and expected use, and describes a solution using new SDP stream attributes plus an optional SIP "hint."

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 January 13, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. Expected Use	5
4. Example Use Cases	5
4.1. Emergency Call from English Speaker in Spain	5
4.2. Emergency Call from Spanish/English Speaker in France	5
4.3. Call to Call Center from Russian Speaker in U.S.	6
4.4. Emergency Call from speech-impaired caller in the U.S.	6
4.5. Emergency Call from deaf caller in the U.S.	6
5. Desired Semantics	7
6. The existing 'lang' attribute	7
7. Proposed Solution	8
7.1. New 'humintlang-send' and 'humintlang-recv' attributes	8
7.2. Advisory vs Required	10
7.3. SIP "hint"	10
7.4. Silly States	11
8. IANA Considerations	11
9. Security Considerations	11
10. Changes from Previous Versions	12
10.1. Changes from draft-gellens-...-00 to -01	12
10.2. Changes from draft-gellens-...-01 to -02	12
10.3. Changes from draft-gellens-...-02 to draft-gellens-mmusic-...-00	12
10.4. draft-gellens-mmusic-...-00 to -01	13
11. Acknowledgments	13
12. References	13
12.1. Normative References	13
12.2. Informational References	14
Author's Address	14

1. Introduction

When setting up interactive communication sessions (using SIP or other protocols), human (natural) language negotiation may be needed. When the caller and callee know each other or where context or out of band information implies the language, such negotiation is typically not needed. In other cases, there is a need for spoken, signed, or written languages to be negotiated based on the caller's preferences and the callee's capabilities. This need applies to both emergency and non-emergency calls. For various reasons, including the ability to establish multiple streams using different media (e.g., voice, text, video), it makes sense to use a per-stream negotiation mechanism, in this case, SDP.

This approach has a number of benefits, including that it is generic (applies to all interactive communications negotiated using SDP) and not limited to emergency calls. In some cases such a facility isn't needed, because the language is known from the context (such as when a caller places a call to a sign language relay center, to a friend, or colleague). But it is clearly useful in many other cases. For example, someone calling a company call center or a Public Safety Answering Point (PSAP) should be able to indicate if one or more specific signed, written, and/or spoken languages are preferred, the callee should be able to indicate its capabilities in this area, and the call proceed using in-common language(s) and media forms.

Since this is a protocol mechanism, the user equipment (UE client) needs to know the user's preferred languages; a reasonable technique could include a configuration mechanism with a default of the language of the user interface. In some cases, a UE could tie language and media preferences, such as a preference for a video stream using a signed language and/or a text or audio stream using a written/spoken language.

Including the user's human (natural) language preferences in the session establishment negotiation is independent of the use of a relay service and is transparent to a voice service provider. For example, assume a user within the United States who speaks Spanish but not English places a voice call using an IMS device. It doesn't matter if the call is an emergency call or not (e.g., to an airline reservation desk). The language information is transparent to the IMS carrier, but is part of the session negotiation between the UE and the terminating entity. In the case of a call to e.g., an airline, the call can be automatically routed to a Spanish-speaking agent. In the case of an emergency call, the Emergency Services IP network (ESInet) and the PSAP may choose to take the language and media preferences into account when determining how to route and process the call (i.e., language and media needs may be considered within policy-based routing (PBR)).

By treating language as another attribute that is negotiated along with other aspects of a media stream, it becomes possible to accommodate a range of users' needs and called party facilities. For example, some users may be able to speak several languages, but have a preference. Some called parties may support some of those languages internally but require the use of a translation service for others, or may have a limited number of call takers able to use certain languages. Another example would be a user who is able to speak but is deaf or hard-of-hearing and requires a voice stream plus a text stream (known as voice carry over). Making language a media attribute allows the standard session negotiation mechanism to handle this by providing the information and mechanism for the endpoints to make appropriate decisions.

Regarding relay services, in the case of an emergency call requiring sign language such as ASL, there are two common approaches: the caller initiates the call to a relay center, or the caller places the call to emergency services (e.g., 911 in the U.S. or 112 in Europe). In the former case, the language need is ancillary and supplemental. In the latter case, the ESInet and/or PSAP may take the need for sign language into account and bridge in a relay center. In this case, the ESInet and PSAP have all the standard information available (such as location) but are able to bridge the relay sooner in the call processing.

By making this facility part of the end-to-end negotiation, the question of which entity provides or engages the relay service becomes separate from the call processing mechanics; if the caller directs the call to a relay service then the human language negotiation facility provides extra information to the relay service but calls will still function without it; if the caller directs the call to emergency services, then the ESInet/PSAP are able to take the user's human language needs into account, e.g., by routing to a particular PSAP or call taker or bridging a relay service or translator.

The term "negotiation" is used here rather than "indication" because human language (spoken/written/signed) is something that can be negotiated in the same way as which forms of media (audio/text/video) or which codecs. For example, if we think of non-emergency calls, such as a user calling an airline reservation center, the user may have a set of languages he or she speaks, with perhaps preferences for one or a few, while the airline reservation center will support a fixed set of languages. Negotiation should select the user's most preferred language that is supported by the call center. Both sides should be aware of which language was negotiated. This is conceptually similar to the way other aspects of each media stream are negotiated using SDP (e.g., media type and codecs).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Expected Use

This facility is expected to be used by NENA and 3GPP. NENA is likely to reference it in NENA 08-01 (i3 Stage 3) in describing attributes of calls presented to an ESInet, and in that or other documents describing Policy-Based Routing (PBR) capabilities within a Policy-Based Routing Function. 3GPP is expected to reference this mechanism in general call handling and emergency call handling. Recent CRs introduced in SA1 have anticipated this functionality being provided within SDP.

4. Example Use Cases

4.1. Emergency Call from English Speaker in Spain

Someone who speaks only English is visiting Spain and places an emergency (112) call. The call offers an audio stream using English. The ESInet and PSAP have policy-based routing rules that take into account the SDP language request when deciding how to route and process the call. The ESInet routes the call to a PSAP within Spain where an English-speaking call taker is available, and the PSAP selects an English-speaking call taker to handle the call. The PSAP answers the offer with an audio stream using English. The call is established with an audio stream; the caller and call taker communicate in English.

Alternatively, the ESInet routes the call to a cooperating PSAP within the U.K. The PSAP answers the offer with an audio stream using English. The call is established with an audio stream; the caller and call taker communicate in English. (This approach is similar to that envisioned in REACH112 Total Conversation.)

4.2. Emergency Call from Spanish/English Speaker in France

Someone who speaks both Spanish and English (but prefers Spanish) is visiting France and places an emergency (112) call. The call offers an audio stream listing first Spanish (meaning most preferred) and then English. The ESInet and PSAP have policy-based routing rules that take into account the SDP language request when deciding how to route and process the call. The ESInet routes the call to a PSAP within France where a Spanish-speaking call taker is available, and the PSAP selects a Spanish-speaking call taker to handle the call. The PSAP answers the offer with an audio stream listing Spanish. The call is established with an audio stream; the caller and call taker

communicate in Spanish.

Alternatively, the ESInet routes the call to a cooperating PSAP in Spain or England. (This approach is similar to that envisioned in REACH112 Total Conversation.)

Alternatively, there is no ESInet or the ESInet does not take language into account in its PBR. The call is routed to a PSAP in France. The PSAP ignores the language information in the SDP offer, and answers the offer with an audio stream with no language or with French. The UE continues the call anyway. The call taker answers in French, the user tries speaking Spanish and perhaps English. The call taker bridges in a translation service or transfers the call to a multilingual call taker.

4.3. Call to Call Center from Russian Speaker in U.S.

A Russian speaker is visiting the U.S. and places a call to her airline reservation desk to inquire about her return flight. The airline call processing system takes into account the SDP language request and decides to route the call to its call center within Russia.

Alternatively, if the airline call processing system does not look at SDP, it uses the SIP "hint" if present.

4.4. Emergency Call from speech-impaired caller in the U.S.

Someone who uses English but is speech-impaired places an emergency (911) call. The call offers an audio stream listing English and a real-time text stream also using English. The ESInet and PSAP have policy-based routing rules that take into account the SDP language and media requests when deciding how to route and process the call. The ESInet routes the call to a PSAP with real-time text capabilities. The PSAP answers the offer with an audio stream listing English and a real-time text stream listing English. The call is established with an audio and a real-time text stream; the caller and call taker communicate in English using voice from the call-taker to the caller and text from the caller to the call taker. The audio stream is two-way, allowing the call taker to hear background sounds.

4.5. Emergency Call from deaf caller in the U.S.

A deaf caller who uses American Sign Language (ASL) places an emergency (911) call. The call offers a video stream listing ASL and an audio stream with no language indicated. The ESInet and PSAP have policy-based routing rules that take into account the SDP language and media needs when deciding how to route and process the call. The ESInet routes the call to a PSAP. The PSAP answers the offer with an

audio stream listing English and a video stream listing ASL. The PSAP bridges in a sign language interpreter. The call is established with an audio and a video stream.

5. Desired Semantics

The desired solution is a media attribute that may be used within an offer to indicate the preferred language of each media stream, and within an answer to indicate the accepted language. The semantics of including multiple values for a media stream within an offer is that the languages are listed in order of preference.

(While it is true that a conversation among multilingual people often involves multiple languages, the usefulness of providing a way to negotiate this as a general facility is outweighed by the complexity of the desired semantics of the SDP attribute to allow negotiation of multiple simultaneous languages within an interactive media stream.)

6. The existing 'lang' attribute

RFC 4566 specifies an attribute 'lang' which sounds similar to what is needed here, the difference being that it specifies that 'a=lang' is declarative with the semantics of multiple 'lang' attributes being that all of them are used, while we want a means to negotiate which one is used in each stream. This difference means that the existing 'lang' attribute can't be used and we need to define a new attribute.

The text from RFC 4566 [RFC4566] is:

```
a=lang:<language tag>
```

This can be a session-level attribute or a media-level attribute. As a session-level attribute, it specifies the default language for the session being described. As a media-level attribute, it specifies the language for that media, overriding any session-level language specified. Multiple lang attributes can be provided either at session or media level if the session description or media use multiple languages, in which case the order of the attributes indicates the order of importance of the various languages in the session or media from most important to least important.

The "lang" attribute value must be a single [RFC3066] language tag in US-ASCII [RFC3066]. It is not dependent on the charset attribute. A "lang" attribute SHOULD be specified when a session is of sufficient scope to cross geographic boundaries where the language of recipients cannot be assumed, or where the session is in a different language from the locally assumed norm.

Note that there are existing examples of it being used in exactly the way we need. For example, draft-saintandre-sip-xmpp-chat-04 [I-D .saintandre-sip-xmpp-chat] contains an example where the initial invitation contains two 'a=lang' entries for a media stream (for English and Italian) and the OK accepts one of them (Italian), which matches what we need:

Example: (F1) SIP user starts the session

```
INVITE sip:juliet@example.com SIP/2.0
To: <sip:juliet@example.com>
From: <sip:romeo@example.net>;tag=576
Subject: Open chat with Romeo?
Call-ID: 742507no
Content-Type: application/sdp

c=IN IP4 s2x.example.net
m=message 7313 TCP/MSRP *
a=accept-types:text/plain
a=lang:en
a=lang:it
a=path:msrp://s2x.example.net:7313/ansp7lweztas;tcp
```

Example: (F2) Gateway accepts session on Juliet's behalf

```
SIP/2.0 200 OK
To: <sip:juliet@example.com>;tag=534
From: <sip:romeo@example.net>;tag=576
Call-ID: 742507no
Content-Type: application/sdp

c=IN IP4 x2s.example.com
m=message 8763 TCP/MSRP *
a=accept-types:text/plain
a=lang:it
a=path:msrp://x2s.example.com:8763/lkjh37s2s20w2a;tcp
```

The example serves as evidence of the need for an SDP attribute with the semantics as described in this document; unfortunately, the existing 'lang' attribute is not it.

7. Proposed Solution

An SDP attribute seems the natural choice to negotiate human (natural) language of an interactive media stream. The attribute value should be a language tag per RFC 5646 [RFC5646]

7.1. New 'humintlang-send' and 'humintlang-recv' attributes

Rather than re-use 'lang' we define two new media-level attributes starting with 'humintlang' (short for "human interactive language") to negotiate which human language is used in each (interactive) media stream. There are two attributes, one ending in "-send" and the other in "-recv" to indicate the language used when sending and receiving media:

a=humintlang-send:<language tag>

a=humintlang-recv:<language tag>

Each can appear multiple times in an offer for a media stream.

In an offer, the 'humintlang-send' values constitute a list in preference order (first is most preferred) of the languages the offerer wishes to send, and the 'humintlang-recv' values constitute a list in preference order of the languages the offerer wishes to receive. In cases where the user wishes to use one media for sending and another for receiving (such as a speech-impaired user who wishes to send using text and receive using audio), one of the two will be unset. In other cases, both SHOULD have the same values in the same order. The two SHOULD NOT be set to languages which are difficult to match together (e.g., specifying a desire to send audio in Hungarian and receive audio in Portuguese will make it difficult to successfully complete the call).

In an answer, 'humintlang-send' is the accepted language the answerer will send (which in most cases is one of the languages in the offer's 'humintlang-recv'), and 'humintlang-recv' is the accepted language the answerer expects to receive (which in most cases is one of the languages in the offer's 'humintlang-send').

Each value MUST be a language tag per RFC 5646 [RFC5646]. RFC 5646 describes mechanisms for matching language tags. While RFC 5646 provides a mechanism accommodating increasingly fine-grained distinctions, in the interest of maximum interoperability for real-time interactive communications, each 'humintlang-send' and 'humintlang-recv' value SHOULD be restricted to the largest granularity of language tags; in other words, it is RECOMMENDED to specify only a Primary-subtag and NOT to include subtags (e.g., for region or dialect) unless the languages might be mutually incomprehensible without them.

In an offer, each language tag value MAY have an asterisk appended as the last character (after the registry value). The asterisk indicates a request by the caller to not fail the call if there is no language in common. See Section 7.2 for more information and discussion.

When placing an emergency call, and in any other case where the language cannot be assumed from context, each media stream in an

offer SHOULD specify one or both 'humintlang-send' and 'humintlang-recv' attributes (to avoid ambiguity).

Note that while signed language tags are used with a video stream to indicate sign language, a spoken language tag for a video stream in parallel with an audio stream with the same spoken language tag indicates a request for a supplemental video stream to see the speaker.

Clients acting on behalf of end users are expected to set one or both 'humintlang-send' and 'humintlang-recv' attributes on each media stream in an offer when placing an outgoing session but ignore the attributes when receiving incoming calls. Systems acting on behalf of call centers and PSAPs are expected to take into account the values when processing inbound calls.

7.2. Advisory vs Required

One important consideration with this mechanism is if the call fails if the callee does not support any of the languages requested by the caller.

In order to provide for maximum likelihood of a successful communication session, especially in the case of emergency calling, the mechanism defined here provides a way for the caller to indicate a preference for the call failing or succeeding when there is no language in common. However, the callee is NOT REQUIRED to honor this preference. For example, a PSAP MAY choose to attempt the call even with no language in common, while a corporate call center MAY choose to fail the call.

The mechanism for indicating this preference is that, in an offer, if the last character of any of the 'humintlang-recv' or 'humintlang-send' values is an asterisk, this indicates a request to not fail the call (similar to SIP Accept-Language syntax). Either way, the called party MAY ignore this, e.g., for the emergency services use case, a PSAP will likely not fail the call.

7.3. SIP "hint"

SDP is used for stream negotiation, and emergency services based on NENA i3 have the ability to reference SDP within policy-based routing rules. However, it is possible that some entities wishing to take the caller's language needs into account may lack this ability. Accordingly, a SIP header is provided to supply a "hint" regarding the caller's language needs. This is merely a hint, not actual negotiation.

Protocols other than SIP may be used to establish interactive communication sessions; this document does not provide a "hint"

mechanism for any protocol other than SIP.

TBD: The SIP header used for the "hint" is either a new header or caller preferences (RFC 3841), presumably Accept-Contact (where the caller sets media and language and optionally required; it's not yet clear if this matches our desired semantics.

This SIP header is just a hint; it is RECOMMENDED to be sent; it is NOT REQUIRED to be sent or to be used on receipt.

In the case of spoken languages using an audio stream, this "hint" regarding language may be sufficient to allow the callee to optimally handle the call, but since the "hint" only deals with language, not media type, it is not sufficient when the caller requests non-audio media such as text or video.

7.4. Silly States

It is possible to specify a "silly state" where the language specified does not make sense for the media type, such as specifying a signed language for an audio media stream.

An offer MUST NOT be created where the language does not make sense for the media type. If such an offer is received, the receiver MAY reject the media, ignore the language specified, or attempt to interpret the intent (e.g., if American Sign Language is specified for an audio media stream, this might be interpreted as a desire to use spoken English).

A spoken language tag for a video stream in conjunction with an audio stream with the same language might indicate a request for supplemental video to see the speaker.

8. IANA Considerations

IANA is kindly requested to add two entries to the 'att-field (media level only)' table of the SDP parameters registry:

Type	Name	Reference
att-field (media level only)	humintlang-send	(this document)
att-field (media level only)	humintlang-recv	(this document)

9. Security Considerations

The Security Considerations of RFC 5646 [RFC5646] apply here (as a use of that RFC). In addition, if the 'humintlang-send' or 'humintlang-recv' values are altered or deleted en route, the session could fail or languages incomprehensible to the caller could be selected; however, this is also a risk if any SDP parameters are modified en route.

10. Changes from Previous Versions

10.1. Changes from draft-gellens-...-00 to -01

- o Changed name of (possible) new attribute from 'humlang' to "humintlang"
- o Added discussion of silly state (language not appropriate for media type)
- o Added Voice Carry Over example
- o Added mention of multilingual people and multiple languages
- o Minor text clarifications

10.2. Changes from draft-gellens-...-01 to -02

- o Updated text for (possible) new attribute "humintlang" to reference RFC 5646
- o Added clarifying text for (possible) re-use of existing 'lang' attribute saying that the registration would be updated to reflect different semantics for multiple values for interactive versus non-interactive media.
- o Added clarifying text for (possible) new attribute "humintlang" to attempt to better describe the role of language tags in media in an offer and an answer.

10.3. Changes from draft-gellens-...-02 to draft-gellens-mmusic-...-00

- o Updated text to refer to RFC 5646 rather than the IANA language subtags registry directly.
- o Moved discussion of existing 'lang' attribute out of "Proposed Solution" section and into own section now that it is not part of proposal.
- o Updated text about existing 'lang' attribute.
- o Added example use cases.
- o Replaced proposed single 'humintlang' attribute with 'humintlang-send' and 'humintlang-recv' per Harald's request/information that it was a misuse of SDP to use the same attribute for sending and receiving.
- o Added section describing usage being advisory vs required and text in attribute section.
- o Added section on SIP "hint" header (not yet nailed down between new and existing header).

- o Added text discussing usage in policy-based routing function or use of SIP header "hint" if unable to do so.
- o Added SHOULD that the value of the parameters stick to the largest granularity of language tags.
- o Added text to Introduction to be try and be more clear about purpose of document and problem being solved.
- o Many wording improvements and clarifications throughout the document.
- o Filled in Security Considerations.
- o Filled in IANA Considerations.
- o Added to Acknowledgments those who participated in the Orlando ad-hoc discussion as well as those who participated in email discussion and side one-on-one discussions.

10.4. draft-gellens-mmusic-...-00 to -01

- o Relaxed language on setting -send and -receive to same values; added text on leaving on empty to indicate asymmetric usage.
- o Added text that clients on behalf of end users are expected to set the attributes on outgoing calls and ignore on incoming calls while systems on behalf of call centers and PSAPs are expected to take the attributes into account when processing incoming calls.

11. Acknowledgments

Many thanks to Bernard Aboba, Harald Alvestrand, Flemming Andreassen, Francois Audet, Eric Burger, Keith Drage, Doug Ewell, Christian Groves, Gunnar Hellstrom, Andrew Hutton, Hadriel Kaplan, Ari Keranen, John Klensin, Paul Kyzivat, John Levine, Alexey Melnikov, James Polk, Pete Resnick, Peter Saint-Andre, and Dale Worley for reviews, corrections, suggestions, and participating in in-person and email discussions.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4566] Handley, M., Jacobson, V. and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

12.2. Informational References

[I-D.iab-privacy-considerations]

Cooper, A., Tschofenig, H., Aboba, B., Peterson, J.,
Morris, J., Hansen, M. and R. Smith, "Privacy
Considerations for Internet Protocols", Internet-Draft
draft-iab-privacy-considerations-03, July 2012.

[I-D.saintandre-sip-xmpp-chat]

Saint-Andre, P., Gavita, E., Hossain, N. and S. Loreto,
"Interworking between the Session Initiation Protocol
(SIP) and the Extensible Messaging and Presence Protocol
(XMPP): One-to-One Text Chat", Internet-Draft draft-
saintandre-sip-xmpp-chat-04, October 2012.

[RFC3066] Alvestrand, H., "Tags for the Identification of
Languages", RFC 3066, January 2001.

Author's Address

Randall Gellens
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, CA 92121
US

Email: rg+ietf@qti.qualcomm.com

MMUSIC Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 16, 2013

C. Holmberg
Ericsson
H. Alvestrand
Google
C. Jennings
Cisco
June 14, 2013

Multiplexing Negotiation Using Session Description Protocol (SDP) Port
Numbers
draft-ietf-mmusic-sdp-bundle-negotiation-04.txt

Abstract

This specification defines a new SDP Grouping Framework extension, "BUNDLE", that can be used with the Session Description Protocol (SDP) Offer/Answer mechanism to negotiate the usage of bundled media, which refers to the usage of a single 5-tuple for media associated with multiple SDP media descriptions ("m=" lines).

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 December 16, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Conventions	4
4. Applicability Statement	5
5. SDP Grouping Framework BUNDLE Extension Semantics	5
6. SDP Offer/Answer Procedures	5
6.1. General	5
6.2. Bundled SDP Information	5
6.2.1. General	5
6.2.2. Bandwidth (b=)	6
6.2.3. rtcp-mux Attribute	6
6.2.4. rtcp Attribute	6
6.2.5. DTLS-SRTP fingerprint Attribute	6
6.2.6. SDES crypto Attribute	6
6.2.7. Other Attributes (a=)	6
6.3. RFC 5888 restrictions	6
6.4. SDP Offerer Procedures	6
6.4.1. SDP Offerer Bundle Address Request and Usage	7
6.4.2. Bundle Address Synchronization (BAS)	7
6.4.3. Adding a media description to a BUNDLE group	8
6.4.4. Moving A Media Description Out Of A BUNDLE Group	8
6.4.5. Disabling A Media Description In A BUNDLE Group	9
6.5. SDP Answerer Procedures	9
6.5.1. Answerer Bundle Address Selection and Usage	9
6.5.2. Moving A Media Description Out Of A BUNDLE Group	10
6.5.3. Rejecting A Media Description In A BUNDLE Group	10
7. Single vs Multiple RTP Sessions	11
7.1. General	11
7.2. Single RTP Session	11
8. Usage With ICE	11
8.1. General	11
8.2. Candidates	11
8.3. Candidates	12
9. Security Considerations	12
10. Examples	12
10.1. Example: Bundle Address Selection	12
10.2. Example: Bundle Mechanism Rejected	14
10.3. Example: Offerer Adds A Media Description To A BUNDLE Group	15
10.4. Example: Offerer Moves A Media Description Out Of A BUNDLE Group	17

10.5. Example: Offerer Disables A Media Description In A BUNDLE Group	18
11. IANA Considerations	19
12. Acknowledgements	19
13. Change Log	20
14. References	20
14.1. Normative References	21
14.2. Informative References	21
Appendix A. Design Considerations	21
A.1. General	21
A.2. UA Interoperability	22
A.3. Usage of port number value zero	23
A.4. B2BUA And Proxy Interoperability	24
A.4.1. Traffic Policing	24
A.4.2. Bandwidth Allocation	25
A.5. Candidate Gathering	25
Authors' Addresses	25

1. Introduction

In the IETF RTCWEB WG, a need to use a single 5-tuple for sending and receiving media associated with multiple SDP media descriptions ("m=" lines) has been identified. This would e.g. allow the usage of a single set of Interactive Connectivity Establishment (ICE) [RFC5245] candidates for multiple media descriptions. Normally different media types (audio, video etc) will be described using different media descriptions.

This specification defines a new SDP Grouping Framework [RFC5888] extension, "BUNDLE", that can be used with the Session Description Protocol (SDP) Offer/Answer mechanism [RFC3264] to negotiate the usage of bundled media, which refers to the usage of a single 5-tuple for media associated with multiple SDP media descriptions ("m=" lines).

The Offerer and Answerer [RFC3264] use the BUNDLE mechanism to negotiate a single BUNDLE address to be used for the bundled media associated with a BUNDLE group.

The BUNDLE mechanism allows an SDP Offerer and SDP Answerer to assign identical addresses to multiple "m=" lines, if those "m=" lines are associated with a BUNDLE group. However, until it is known whether both the Offerer and Answerer support the BUNDLE mechanism, unique addresses are assigned to each "m=" line, including those associated with a BUNDLE group.

NOTE: As defined in RFC 4566 [RFC4566], the semantics of multiple "m=" lines using the same port number value are undefined, and there

is no grouping defined by such means. Instead, an explicit grouping mechanism needs to be used to express the intended semantics. This specification provides such extension.

SDP Offers and SDP Answer can contain multiple BUNDLE groups. For each BUNDLE group, a BUNDLE address is negotiated. Multiple BUNDLE groups cannot share the same bundle address.

The default assumption is that all Real-Time Protocol (RTP) [RFC3550] based media flows within a BUNDLE group belongs to the same RTP Session [RFC3550]. Future extensions can change that assumption.

The BUNDLE mechanism is backward compatible. Endpoints that do not support the BUNDLE mechanism are expected to generate SDP Offers and SDP Answers without an SDP group:BUNDLE attribute, and are expected to assign unique addresses to each "m=" line, according to the procedures in [RFC4566] and [RFC3264]

2. Terminology

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

Bundled media: Two or more RTP streams using a single 5-tuple. The RTCP streams associated with the RTP streams also use a single 5-tuple, which might be the same, but can also be different, as the one used by the RTP streams.

Unique address: This refers to an IP address and IP port combination, that can only be associated with a single "m=" line within an SDP Session.

BUNDLE address: This refers to an IP address and IP port combination, that is associated with each "m=" line within a BUNDLE group, within an SDP Session. The zero IP port value BUNDLE address MUST NOT be used in a BUNDLE address.

NOTE: "m=" lines that share a BUNDLE address MUST also share other parameters related to the media transport plane, e.g. ICE candidate information.

3. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

4. Applicability Statement

The mechanism in this specification only applies to the Session Description Protocol (SDP) [RFC4566], when used together with the SDP Offer/Answer mechanism [RFC3264].

5. SDP Grouping Framework BUNDLE Extension Semantics

This section defines a new SDP Grouping Framework extension, BUNDLE.

The BUNDLE extension can be indicated using an SDP session-level 'group' attribute. Each SDP Media Description ("m=" line) that is grouped together, using SDP media-level mid attributes, belongs to a given BUNDLE group.

6. SDP Offer/Answer Procedures

6.1. General

This section describes the usage of the SDP Offer/Answer mechanism [RFC3264] to negotiate the usage of the BUNDLE mechanism, to negotiate the BUNDLE address, and to add, remove and reject SDP Media Descriptions ("m=" lines) [RFC4566] associated with a BUNDLE group.

The generic rules and procedures defined in [RFC3264] and [RFC5888] apply when the SDP Offer/Answer mechanism is used with the BUNDLE mechanism. For example, if an SDP Offer is rejected, the previously negotiated SDP parameters and characteristics (including those associated with BUNDLE groups) apply.

When an endpoint, acting as an Offerer or Answerer [RFC3264], generates an SDP Offer, or an SDP Answer, the endpoint MUST assign an SDP media-level mid value for each "m=" line in a BUNDLE group. In addition, the endpoint MUST assign an SDP session-level group:BUNDLE attribute for each BUNDLE group, and place each mid associated with the SDP group:BUNDLE attribute mid list.

6.2. Bundled SDP Information

6.2.1. General

This section describes restrictions associated with the usage of SDP parameters and extensions within a BUNDLE group. It also describes, when parameter and attribute values have been assigned to each "m=" line in the BUNDLE group, how to calculate a value for the whole BUNDLE group.

6.2.2. Bandwidth (b=)

The total proposed bandwidth is the sum of the proposed bandwidth for each "m=" line associated with a negotiated BUNDLE group.

6.2.3. rtcp-mux Attribute

For each "m=" line in a BUNDLE group, an Offerer and Answerer MUST assign an SDP rtcp-mux attribute [RFC5761].

6.2.4. rtcp Attribute

When used, for each RTP media "m=" line in a BUNDLE group, an Offerer and Answerer MUST assign an SDP rtcp attribute [RFC3605] with an identical attribute value.

6.2.5. DTLS-SRTP fingerprint Attribute

When DTLS-SRTP is used, for each RTP media "m=" line in a BUNDLE group, an Offerer and Answerer MUST assign an SDP DTLS-SRTP fingerprint attribute with identical attribute values.

6.2.6. SDES crypto Attribute

When SDES is used, for each RTP media "m=" line in a BUNDLE group, an Offerer and Answerer MUST assign an SDP crypto attribute, with unique attribute values.

6.2.7. Other Attributes (a=)

There are also special rules for handling many different attributes as defined in [I-D.nandakumar-mmusic-sdp-attributes]. It might not be possible to use bundle with some attributes.

6.3. RFC 5888 restrictions

Based on the rules and procedures in [RFC5888], the following restrictions also apply to BUNDLE groups in SDP Answers:

- o 1) A BUNDLE group must not be added to an SDP Answer, unless the same BUNDLE group was included in the associated SDP Offer; and
- o 2) An SDP "m=" line must not be added to a BUNDLE group in the SDP Answer, unless it was in the same BUNDLE group in the associated SDP Offer.

6.4. SDP Offerer Procedures

6.4.1. SDP Offerer Bundle Address Request and Usage

An Offerer can assign a BUNDLE address to multiple "m=" lines in a BUNDLE group, once the Answerer has selected the BUNDLE address for the Offerer. An Offerer **MUST NOT** assign a BUNDLE address to multiple "m=" lines until the Answerer has selected the BUNDLE address.

OPEN ISSUE: Should it be allowed to assign a new BUNDLE address to multiple "m=" lines in a BUNDLE group, before the Answerer has selected the BUNDLE address?

In order to negotiate (or, to re-negotiate) the BUNDLE address associated with a BUNDLE group, the Offerer, in the SDP Offer, assigns a unique address to each "m=" line in the BUNDLE group. In addition, the Offerer indicates which unique address it wishes the Answerer to select as the Offerer's BUNDLE address. The Offerer places the mid, associated with the unique address requested to be selected as the BUNDLE address, first in the SDP group:BUNDLE attribute mid list. The Answerer will then select the BUNDLE address for the Offerer ([ref-to-be-added]).

If the Offerer, in a subsequent SDP Offer, wants to re-negotiate the BUNDLE address associated with a BUNDLE group, it **MAY** assign the previously negotiated BUNDLE address as a unique address to one of the "m=" lines in the BUNDLE group.

If the Offerer assigns the previously selected BUNDLE address to more than one "m=" line in a BUNDLE group, the first mid in the SDP group:BUNDLE attribute mid list **MUST** represent an "m=" line to which the BUNDLE address is assigned. Hence, in order to re-negotiate the BUNDLE address, the Offerer needs to assign a unique address to each "m=" line in the BUNDLE group, as described above.

An Offerer **MUST NOT** assign a BUNDLE address to an "m=" line that is not associated with a BUNDLE group. An Offerer **MUST NOT** assign a BUNDLE address, that has been negotiated for a BUNDLE group, to an "m=" line that is associated with another BUNDLE group.

Section 10.1 shows an example of a Bundle Address Request.

6.4.2. Bundle Address Synchronization (BAS)

When an Offerer has requested the Answerer to select the Offerer's BUNDLE address, and the Offerer did not assign the requested BUNDLE address to each "m=" line in the BUNDLE group of the SDP Offer used to request the BUNDLE address, when the associated SDP Answer is received the Offerer **MUST** send a subsequent SDP Offer. In the subsequent SDP Offer the Offerer **MUST** assign the selected BUNDLE

address to each "m=" line in the BUNDLE group. This procedure is referred to as Bundle Address Synchronization (BAS).

When the Offerer performs a BAS, the Offerer MAY modify SDP parameters in the same SDP Offer.

NOTE: It is important that the SDP Offer used for the BAS gets accepted by the Answerer, so the Offerer needs to consider the necessity to modify SDP parameters that could get the Answerer to reject the SDP Offer. Removing "m=" lines, or reducing the number of codecs, in the SDP Offer used for the BAS is considered to have a low risk of being rejected.

NOTE: The main purpose of the BAS is to make sure that intermediaries, that might not support the BUNDLE mechanism, have correct information regarding which IP address and port is going to be used for the bundled media.

Section 10.1 shows an example of an SDP Offer used to perform a BAS.

6.4.3. Adding a media description to a BUNDLE group

When an Offerer adds an "m=" line to a BUNDLE group, the Offerer MUST assign either a unique address, or the BUNDLE address associated with the BUNDLE group, to the added "m=" line. In addition, the Offerer MUST assign a mid value to the "m=" line, and place the mid in the SDP group:BUNDLE attribute mid list associated with the BUNDLE group, in order to group the "m=" line to the BUNDLE group.

NOTE: If the Offerer assigns a unique address to the added "m=" line, it allows the Answerer to move the "m=" line out of the BUNDLE group without having to reject the "m=" line ([ref-to-be-added]).

To the previously added "m=" lines in the BUNDLE group, the Offerer assigns either unique addresses or the BUNDLE address associated with the BUNDLE group, according to the procedures in Section 6.4.1.

Section 10.3 shows an example of an SDP Offer used to add an "m=" line to a BUNDLE group.

6.4.4. Moving A Media Description Out Of A BUNDLE Group

When an Offerer moves an "m=" line out of a BUNDLE group, the Offerer MUST assign a unique address to the moved "m=" line. In addition, the Offerer MUST NOT anymore use a mid value to group the "m=" line with the BUNDLE group.

To the remaining "m=" lines in the BUNDLE group, the Offerer assigns either unique addresses or the BUNDLE address associated with the BUNDLE group, according to the procedures in Section 6.4.1.

Section 10.4 shows an example of an SDP Offer used to move an "m=" line out of a BUNDLE group.

6.4.5. Disabling A Media Description In A BUNDLE Group

When an Offerer disables an "m=" line in a BUNDLE group, the Offerer MUST assign a zero port value [RFC3264] to the disabled "m=" line. In addition, the Offerer MUST NOT anymore use a mid value to group the "m=" line with the BUNDLE group.

To the remaining "m=" lines in the BUNDLE group, the Offerer assigns either unique addresses or the BUNDLE address associated with the BUNDLE group, according to the procedures in Section 6.4.1.

Section 10.5 shows an example of an SDP Offer used to move an "m=" line out of a BUNDLE group.

6.5. SDP Answerer Procedures

6.5.1. Answerer Bundle Address Selection and Usage

6.5.1.1. Offerer Bundle Address Selection

If the Offerer, in an SDP Offer, assigned a unique address to each "m=" line in a BUNDLE group, it means that the Offerer has requested the Answerer to select a BUNDLE address for the Offerer. The first mid in the SDP group:BUNDLE attribute mid list of the SDP Offer represents the unique address which the Offerer requests the Answerer to select as the Offerer's BUNDLE address.

The Answerer SHOULD select the unique address associated with the first mid to become the Offerer's BUNDLE address, unless the Answerer in the SDP Answer will move the "m=" line represented by the mid out of the BUNDLE group, or if there is some other reason why the Answerer can not select the unique address associated with the mid. In that case, the Answerer MUST try the next mid in the list.

In the SDP Answer, the Answerer MUST place the mid associated with the selected BUNDLE address first in the SDP group:BUNDLE attribute mid list associated with the BUNDLE group.

Section 10.1 shows an example of an Offerer's BUNDLE address selection.

6.5.1.2. Answerer Bundle Address Selection

The Answerer MUST select a local BUNDLE address, and in the SDP Answer assign it to each "m=" line associated with the BUNDLE group.

The Answerer is allowed to change its BUNDLE address in any SDP Answer.

The Answerer MUST NOT assign a BUNDLE address to an "m=" line that is not associated with a BUNDLE group. The Answerer MUST NOT assign a BUNDLE address, associated with a BUNDLE group, to an "m=" line associated with another BUNDLE group.

Section 10.1 shows an example of an Answerer's local BUNDLE address selection.

6.5.2. Moving A Media Description Out Of A BUNDLE Group

When an Answerer moves an "m=" line out of a BUNDLE group, the Answerer MUST assign a unique address to the moved "m=" line. In addition, the Answerer MUST NOT anymore use a mid value to group the "m=" line with the BUNDLE group.

To the remaining "m=" lines in the BUNDLE group, the Answerer assigns the Answerer's BUNDLE address.

An Answerer MUST NOT move an "m=" line out of a BUNDLE group, unless:

- o 1) The Offerer assigned a unique address to the "m=" line in the associated SDP Offer; or
- o 2) The Answerer also rejects the "m=" line Section 6.5.3.

6.5.3. Rejecting A Media Description In A BUNDLE Group

When an Answerer rejects an "m=" line in a BUNDLE group, the Answerer MUST assign a zero port value to the rejected "m=" line. In addition, the Answerer MUST NOT anymore use a mid value to group the "m=" line with the BUNDLE group.

To the remaining "m=" lines in the BUNDLE group, the Answerer assigns the Answerer's BUNDLE address.

7. Single vs Multiple RTP Sessions

7.1. General

By default, all RTP based media flows within a given BUNDLE group belong to a single RTP session [RFC3550]. Multiple BUNDLE groups will form multiple RTP Sessions.

The usage of multiple RTP Sessions within a given BUNDLE group, or the usage of a single RTP Session that spans over multiple BUNDLE groups, is outside the scope of this specification. Other specification needs to extend the BUNDLE mechanism in order to allow such usages.

7.2. Single RTP Session

When a single RTP Session is used, media associated with all "m=" lines part of a bundle group share a single SSRC [RFC3550] numbering space.

In addition, the following rules and restrictions apply for a single RTP Session:

- o - The dynamic payload type values used in the "m=" lines MUST NOT overlap.
- o - The "proto" value in each "m=" line MUST be identical (e.g. RTP/AVPF).
- o - A given SSRC SHOULD NOT transmit RTP packets using payload types that originates from different "m=" lines.

NOTE: The last bullet above is to avoid sending multiple media types from the same SSRC. If transmission of multiple media types are done with time overlap RTP and RTCP fails to function. Even if done in proper sequence this causes RTP Timestamp rate switching issues [ref to draft-ietf-avtext-multiple-clock-rates].

8. Usage With ICE

8.1. General

This section describes how to use the BUNDLE grouping extension together with the Interactive Connectivity Establishment (ICE) mechanism [RFC5245].

8.2. Candidates

When an ICE-enabled endpoint generates an SDP Offer, which contains a BUNDLE group, the SDP Offerer MUST include ICE candidates for each "m=" line associated with a "BUNDLE" group, except for any "m=" line with a zero port number value. If the "m=" lines associated with the BUNDLE group contain different port number values, the SDP Offerer MUST also insert different candidate values in each "m=" line associated with the BUNDLE group. If the "m=" lines associated with the BUNDLE group contain an identical port number value, the candidate values MUST also be identical.

When an ICE-enabled endpoint generates an SDP Answer, which contains a BUNDLE group, the Answerer MUST include ICE candidates for each "m=" line associated with the "BUNDLE" group, except for any "m=" line where the port number value is set to zero. The Answerer MUST insert identical candidate values in each "m=" line associated with the BUNDLE group.

8.3. Candidates

Once it is known that both endpoints support, and accept to use, the BUNDLE grouping extension, ICE connectivity checks and keep-alives only needs to be performed for the whole BUNDLE group, instead of for each individual "m=" line associated with the group.

9. Security Considerations

This specification does not significantly change the security considerations of SDP which can be found in Section X of TBD.

TODO: Think carefully about security analysis of reuse of same SDP key on multiple "m=" lines when the far end does not use BUNDLE and warn developers of any risks.

10. Examples

10.1. Example: Bundle Address Selection

The example below shows:

- o 1. An SDP Offer, in which the Offerer assigns unique addresses to each "m=" line in the BUNDLE group, and requests the Answerer to select the Offerer's BUNDLE address.
- o 2. An SDP Answer, in which the Answerer selects the BUNDLE address for the Offerer, and assigns its own local BUNDLE address to each "m=" line in the BUNDLE group.

- o 3. A subsequent SDP Offer, which is used to perform a Bundle Address Synchronization (BAS).

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 10002 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.com
s=
c=IN IP4 host.biloxi.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
```

SDP Offer (3)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
```

```
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 10000 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

10.2. Example: Bundle Mechanism Rejected

The example below shows:

- o 1. An SDP Offer, in which the Offerer assigns unique addresses to each "m=" line in the BUNDLE group, and requests the Answerer to select the Offerer's BUNDLE address.
- o 2. An SDP Answer, in which the Answerer rejects the BUNDLE group, and assigns unique addresses to each "m=" line.

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 10002 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.com
s=
c=IN IP4 host.biloxi.com
t=0 0
m=audio 20000 RTP/AVP 0
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 30000 RTP/AVP 32
b=AS:1000
a=rtpmap:32 MPV/90000
```

10.3. Example: Offerer Adds A Media Description To A BUNDLE Group

The example below shows:

- o 1. An SDP Offer, in which the Offerer adds an "m=" line, represented by the "zen" mid value, to a previously negotiated BUNDLE group, assigns a unique address to the added "m=" line, and assigns the previously negotiated BUNDLE address to the previously added "m=" lines in the BUNDLE group.
- o 2. An SDP Answer, in which the Answerer assigns its own local BUNDLE address to each "m=" line (including the added "m=" line) in the BUNDLE group.
- o 3. A subsequent SDP Offer, which is used to perform a Bundle Address Synchronization (BAS).

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar zen
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```



```
a=rtpmap:97 iLBC/8000
m=video 10000 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
m=video 20000 RTP/AVP 66
a=mid:zen
b=AS:1000
a=rtpmap:66 H261/90000
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.com
s=
c=IN IP4 host.biloxi.com
t=0 0
a=group:BUNDLE foo bar zen
m=audio 20000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
m=video 20000 RTP/AVP 66
a=mid:zen
b=AS:1000
a=rtpmap:66 H261/90000
```

SDP Offer (3)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar zen
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
```

```
m=video 10000 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
m=video 10000 RTP/AVP 66
a=mid:zen
b=AS:1000
a=rtpmap:66 H261/90000
```

10.4. Example: Offerer Moves A Media Description Out Of A BUNDLE Group

The example below shows:

- o 1. An SDP Offer, in which the Offerer moves an "m=" line out of a previously negotiated BUNDLE group, assigns a unique address to the moved "m=" line, and assigns the previously negotiated BUNDLE address to the remaining "m=" lines in the BUNDLE group.
- o 2. An SDP Answer, in which the Answerer moves the corresponding "m=" line out of the BUNDLE group, and assigns unique address to the moved "m=" line, and assigns the previously negotiated BUNDLE address to the remaining "m=" lines in the BUNDLE group.

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 10000 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
m=video 50000 RTP/AVP 66
```

```
b=AS:1000
a=rtpmap:66 H261/90000
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.com
s=
c=IN IP4 host.biloxi.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
m=video 60000 RTP/AVP 66
b=AS:1000
a=rtpmap:66 H261/90000
```

10.5. Example: Offerer Disables A Media Description In A BUNDLE Group

The example below shows:

- o 1. An SDP Offer, in which the Offerer moves an "m=" line out of a previously negotiated BUNDLE group, assigns a zero port number the moved "m=" line in order to disable it, and assigns the previously negotiated BUNDLE address to the remaining "m=" lines in the BUNDLE group.
- o 2. An SDP Answer, in which the Answerer moves the corresponding "m=" line out of the BUNDLE group, and assigns a zero port value to the moved "m=" line in order to disable it, and assigns the previously negotiated BUNDLE address to the remaining "m=" lines in the BUNDLE group.

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
```

```
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 10000 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
m=video 0 RTP/AVP 66
a=rtpmap:66 H261/90000
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.com
s=
c=IN IP4 host.biloxi.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
m=video 0 RTP/AVP 66
a=rtpmap:66 H261/90000
```

11. IANA Considerations

This document requests IANA to register the new SDP Grouping semantic extension called BUNDLE.

12. Acknowledgements

The usage of the SDP grouping extension for negotiating bundled media is based on a similar alternatives proposed by Harald Alvestrand and Cullen Jennings. The BUNDLE mechanism described in this document is based on the different alternative proposals, and text (e.g. SDP examples) have been borrowed (and, in some cases, modified) from those alternative proposals.

The SDP examples are also modified versions from the ones in the Alvestrand proposal.

Thanks to Paul Kyzivat and Martin Thompson for taking the the time to read the text along the way, and providing useful feedback.

13. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-02

- o Mechanism modified, to be based on usage of SDP Offers with both different and identical port number values, depending on whether it is known if the remote endpoint supports the extension.
- o Cullen Jennings added as co-author.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-01

- o No changes. New version due to expiration.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-00

- o No changes. New version due to expiration.

Changes from draft-holmberg-mmusic-sdp-multiplex-negotiation-00

- o Draft name changed.
- o Harald Alvestrand added as co-author.
- o "Multiplex" terminology changed to "bundle".
- o Added text about single versus multiple RTP Sessions.
- o Added reference to RFC 3550.

14. References

14.1. 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.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [I-D.nandakumar-mmusic-sdp-attributes] Nandakumar, S. and C. Jennings, "A Framework for SDP Attributes when Multiplexing", draft-nandakumar-mmusic-sdp-attributes-00 (work in progress), February 2013.

14.2. Informative References

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

Appendix A. Design Considerations

A.1. General

One of the main issues regarding the BUNDLE grouping extensions has been whether, in SDP Offers and SDP Answers, the same port number value should be inserted in "m=" lines associated with a BUNDLE group, as the purpose of the extension is to negotiate the usage of a single 5-tuple for media associated with the "m=" lines. Issues with both approaches, discussed in the Appendix have been raised. The

outcome was to specify a mechanism which uses SDP Offers with both different and identical port number values.

Below are the primary issues that have been considered when defining the "BUNDLE" grouping extension:

- o 1) Interoperability with existing UAs.
- o 2) Interoperability with intermediary B2BUA- and proxy entities.
- o 3) Time to gather, and the number of, ICE candidates.
- o 4) Different error scenarios, and when they occur.
- o 5) SDP Offer/Answer impacts, including usage of port number value zero.

NOTE: Before this document is published as an RFC, this Appendix might be removed.

A.2. UA Interoperability

Consider the following SDP Offer/Answer exchange, where Alice sends an SDP Offer to Bob:

SDP Offer

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
m=audio 10000 RTP/AVP 97
a=rtpmap:97 iLBC/8000
m=video 10002 RTP/AVP 97
a=rtpmap:97 H261/90000
```

SDP Answer

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

```
t=0 0
m=audio 20000 RTP/AVP 97
a=rtpmap:97 iLBC/8000
m=video 20002 RTP/AVP 97
a=rtpmap:97 H261/90000
```

RFC 4961 specifies a way of doing symmetric RTP but that is an a later invention to RTP and Bob can not assume that Alice supports RFC 4961. This means that Alice may be sending RTP from a different port than 10000 or 10002 - some implementation simply send the RTP from an ephemeral port. When Bob's endpoint receives an RTP packet, the only way that Bob know if it should be passed to the video or audio codec is by looking at the port it was received on. This lead some SDP implementations to use the fact that each "m=" line had a different port number to use that port number as an index to find the correct m line in the SDP. As a result, some implementations that do support symmetric RTP and ICE still use a SDP data structure where SDP with "m=" lines with the same port such as:

SDP Offer

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
m=audio 10000 RTP/AVP 97
a=rtpmap:97 iLBC/8000
m=video 10000 RTP/AVP 98
a=rtpmap:98 H261/90000
```

will result in the second "m=" line being considered an SDP error because it has the same port as the first line.

A.3. Usage of port number value zero

In an SDP Offer or SDP Answer, the media associated with an "m=" line can be disabled/rejected by setting the port number value to zero. This is different from e.g. using the SDP direction attributes, where RTCP traffic will continue even if the SDP "inactive" attribute is indicated for the associated "m=" line.

If each "m=" line associated with a BUNDLE group would contain different port number values, and one of those port would be used for the 5-tuple, problems would occur if an endpoint wants to disable/reject the "m=" line associated with that port, by setting the port number value to zero. After that, no "m=" line would contain the port number value which is used for the 5-tuple. In addition, it is unclear what would happen to the ICE candidates associated with the "m=" line, as they are also used for the 5-tuple.

A.4. B2BUA And Proxy Interoperability

Some back to back user agents may be configured in a mode where if the incoming call leg contains an SDP attribute the B2BUA does not understand, the B2BUA still generates that SDP attribute in the Offer for the outgoing call leg. Consider an B2BUA that did not understand the SDP "rtcp" attribute, defined in RFC 3605, yet acted this way. Further assume that the B2BUA was configured to tear down any call where it did not see any RTCP for 5 minutes. In this cases, if the B2BUA received an Offer like:

SDP Offer

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtcp:53020
```

It would be looking for RTCP on port 49172 but would not see any because the RTCP would be on port 53020 and after five minutes, it would tear down the call. Similarly, an SBC that did not understand BUNDLE yet put BUNDLE in it's offer may be looking for media on the wrong port and tear down the call. It is worth noting that a B2BUA that generated an Offer with capabilities it does not understand is not compliant with the specifications.

A.4.1. Traffic Policing

Sometimes intermediaries do not act as B2BUA, in the sense that they don't modify SDP bodies, nor do they terminate SIP dialogs. Still, however, they may use SDP information (e.g. IP address and port) in order to control traffic gating functions, and to set traffic

policing rules. There might be rules which will trigger a session to be terminated in case media is not sent or received on the ports retrieved from the SDP. This typically occurs once the session is already established and ongoing.

A.4.2. Bandwidth Allocation

Sometimes intermediaries do not act as B2BUA, in the sense that they don't modify SDP bodies, nor do they terminate SIP dialogs. Still, however, they may use SDP information (e.g. codecs and media types) in order to control bandwidth allocation functions. The bandwidth allocation is done per "m=" line, which means that it might not be enough if media associated with all "m=" lines try to use that bandwidth. That may either simply lead to bad user experience, or to termination of the call.

A.5. Candidate Gathering

When using ICE, an candidate needs to be gathered for each port. This takes approximately 20 ms extra for each extra "m=" line due to the NAT pacing requirements. All of this gather can be overlapped with other things while the page is loading to minimize the impact. If the client only wants to generate TURN or STUN ICE candidates for one of the "m=" lines and then use trickle ICE [TODO REF] to get the non host ICE candidates for the rest of the "m=" lines, it MAY do that and will not need any additional gathering time.

Some people have suggested a TURN extension to get a bunch of TURN allocation at once. This would only provide a single STUN result so in cases where the other end did not support BUNDLE, may cause more use of the TURN server but would be quick in the cases where both sides supported BUNDLE and would fall back to a successful call in the other cases.

Authors' Addresses

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Harald Tveit Alvestrand
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

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

Email: fluffy@iii.ca

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

E. Iovov
Jitsi
July 15, 2013

Economical Use of the Offer/Answer Model in Sessions with Multiple Media
Sources
draft-iovov-mmusic-multiple-sources-00

Abstract

The Session Description Protocol (SDP) Offer/Answer model describes a mechanism that allows two entities to negotiate a multimedia session. The SDP syntax of the Offer/Answer model uses constructs known as media (m=) lines to describe each medium. In SDP itself these "m=" lines were designed to describe RTP sessions with any number of streams (SSRCs). Yet, Offer/Answer implementations in SIP applications have most often used them as an envelope for a maximum of two RTP streams (SSRCs) at a time: one in each direction. The most common reason for this has been the fact these applications could not meaningfully render multiple SSRCs simultaneously.

The above situation has led to difficulties once the need to represent multiple (SSRCs) in an interoperable manner became more common.

This document explores the use of "m=" lines for the negotiation of sessions with multiple media sources, as per their original design in SDP. It presents the advantages of such an approach as well as the challenges that it implies in terms of interoperability with already deployed legacy devices.

The model described here was first presented in the RTCWEB No Plan proposal. The reason to spin it off into this new document is mainly to separate the parts related to Offer/Answer and "m=" line semantics, from those that are specific to WebRTC.

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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Mechanism	4
2.1. Discovery	7
2.2. Advantages	8
3. Additional Session Control and Signalling	8
4. Demultiplexing and Identifying Streams (Use of Bundle)	9
5. Simulcasting, FEC, Layering and RTX (Open Issue)	10
6. Problems with Plans A and B	11
7. IANA Considerations	12
8. Informative References	13
Appendix A. Acknowledgements	14
Author's Address	15

1. Introduction

Since its early days the Session Description Protocol (SDP) Offer/Answer (O/A) model [RFC3264] has mainly been used for the negotiation of Real-time Transport Protocol (RTP) [RFC3550] sessions between endpoints that only use a single media source per medium. Examples of such sources include microphones, cameras, desktop streamers etc. The list can be extended to cases where multiple sources are mixed at the media level by an audio or video mixer and then appear as a single stream, i.e. RTP Synchronisation Source (SSRC) on the RTP and SDP [RFC4566] levels.

In such sessions each medium and its corresponding device are described by the SDP "m=" line construct and each media source is mapped to exactly one "m=" line. The exchanges that lead to the establishment of such sessions are relatively well covered by the specifications and most implementations.

Unfortunately, the situation becomes relatively confusing when it comes to transporting multiple media sources (SSRCs) per medium. Streaming any number of RTP streams is an inherent part of the protocol and describing such multi-stream RTP sessions is directly supported by SDP. Still, the Offer/Answer model [RFC3264] is relatively vague on the subject and relying on the multi-stream capabilities of an SDP "m=" line is likely to lead to unexpected results with most endpoints.

At the time of writing of this document, the MMUSIC working group is considering two approaches to addressing the issue. The approaches emerged in the RTCWEB working group and are often referred to as Plan A [PlanA] and Plan B [PlanB]. Both of them impose semantics and syntax that allow for detailed description and fine-grained control of multiple media sources entirely through SDP and Offer/Answer.

Both plans A and B present a number of problems most of which are due to the heavy reliance on SDP O/A. The problems are discussed in more detail in Section 6.

The goal of this document is to propose an alternative approach that simply uses "m=" lines in the way they were originally designed with SDP: descriptors of RTP sessions with any number of sources. Such an approach keeps the use of SDP Offer/Answer to the initialization of transport and media chains and delegates stream control to other, upper layer protocols.

The model described in this specification is intended for applications that require reliability, flexibility and scalability. It therefore tries to satisfy the following constraints

- o the addition and removal of media sources (e.g. conference participants, multiple web cams or "slides") must be possible without the need of Offer/Answer exchanges;
- o the addition or removal of simulcast or layered streams must be possible without the need for Offer/Answer exchanges beyond the initial declaration of such capabilities for either direction.
- o call establishment must not require preliminary announcement or even knowledge of all potentially participating media sources;
- o application specific signalling should be used to cover most semantics following call establishment, such as adding, removing or identifying SSRCs;
- o straightforward interoperability with widely deployed legacy endpoints with rudimentary support for Offer/Answer. This includes devices that allow for one audio and potentially one video m= line and that expect to only ever be required to render a single RTP stream at a time for any of them. (Note that this does NOT include devices that expect to see multiple "m=video" lines for different SSRCs as they can hardly be viewed as "widely deployed legacy").

To achieve the above requirements this specification expects that endpoints will only use SDP Offer/Answer to establish transport channels and initialize an RTP stack and codec/processing chains. This also includes any renegotiation that requires the re-initialisation of these chains. For example, adding VP8 to a session that was setup with only H.264, would obviously still require an Offer/Answer exchange.

All other session control and signalling are to be left to upper layer protocol mechanisms.

2. Mechanism

The model presented in this specification relies on use of standard SDP and Offer/Answer for negotiating formats, establishing transport channels and exchanging, in a declarative way, media and transport parameters that are then used for the initialization of the corresponding stacks. It does not add new concepts and simply requires applications to abide by the original design of SDP and the "m=" line construct.

The following is an example presenting what this specification views as a typical offer sent by a multistream endpoint following this specification:

```
v=0
o=- 0 0 IN IP4 198.51.100.33
s=
t=0 0

a=group:BUNDLE audio video           // declaring BUNDLE Support
c=IN IP4 198.51.100.33
a=ice-ufrag:Qq8o/jZwknkmXpIh         // initializing ICE
a=ice-pwd:gTMACiJcZv1xdPrjfbTHL5qo
a=ice-options:trickle
a=fingerprint:sha-1                  // DTLS-SRTP keying
    a4:b1:97:ab:c7:12:9b:02:12:b8:47:45:df:d8:3a:97:54:08:3f:16

m=audio 5000 RTP/SAVPF 96 0 8
a=mid:audio
a=rtcp-mux

a=rtpmap:96 opus/48000/2              // PT mappings
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000

a=extmap:1 urn:ietf:params:rtp-hdrext:csrc-audio-level //5825 header
a=extmap:2 urn:ietf:params:rtp-hdrext:ssrc-audio-level //extensions

[ICE Candidates]

m=video 5002 RTP/SAVPF 97 98
a=mid:video
a=rtcp-mux

a=rtpmap:97 VP8/90000                // PT mappings and resolutions capabilities
a=imageattr:97 \
    send [x=[480:16:800],y=[320:16:640],par=[1.2-1.3],q=0.6] \
        [x=[176:8:208],y=[144:8:176],par=[1.2-1.3]] \
    recv *
a=rtpmap:98 H264/90000
a=imageattr:98 send [x=800,y=640,sar=1.1,q=0.6] [x=480,y=320] \
    recv [x=330,y=250]

a=extmap:3 urn:ietf:params:rtp-hdrext:fec-source-ssrc //5825 header
a=extmap:4 urn:ietf:params:rtp-hdrext:rtx-source-ssrc //extensions

a=max-send-ssrc:{*:1}                // declaring maximum
a=max-recv-ssrc:{*:4}                // number of SSRCs

[ICE Candidates]
```


The answer to the offer above would have substantially the same structure and content. For the sake of clarity:

```
v=0
o=- 0 0 IN IP4 203.0.113.12
s=
t=0 0

a=group:BUNDLE audio video           // declaring BUNDLE Support
c=IN IP4 203.0.113.12
a=ice-ufrag:Qq8o/jZwknkmXpIh        // initializing ICE
a=ice-pwd:gTMACiJcZvLxdPrjfbTHL5qo
a=ice-options:trickle
a=fingerprint:sha-1                 // DTLS-SRTP keying
    a4:b1:97:ab:c7:12:9b:02:12:b8:47:45:df:d8:3a:97:54:08:3f:16

m=audio 5000 RTP/SAVPF 96 0 8
a=mid:audio
a=rtcp-mux

a=rtpmap:96 opus/48000/2             // PT mappings
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000

a=extmap:1 urn:ietf:params:rtp-hdrext:csrc-audio-level //5825 header
a=extmap:2 urn:ietf:params:rtp-hdrext:ssrc-audio-level //extensions

[ICE Candidates]

m=video 5002 RTP/SAVPF 97 98
a=mid:video
a=rtcp-mux

a=rtpmap:97 VP8/90000               // PT mappings and resolutions capabilities
a=imageattr:97 \
    send [x=[480:16:800],y=[320:16:640],par=[1.2-1.3],q=0.6] \
        [x=[176:8:208],y=[144:8:176],par=[1.2-1.3]] \
    recv *
a=rtpmap:98 H264/90000
a=imageattr:98 send [x=800,y=640,sar=1.1,q=0.6] [x=480,y=320] \
    recv [x=330,y=250]

a=extmap:3 urn:ietf:params:rtp-hdrext:fec-source-ssrc //5825 header
a=extmap:4 urn:ietf:params:rtp-hdrext:rtx-source-ssrc //extensions

a=max-send-ssrc:{*:4}              // declaring maximum
```

```
a=max-recv-ssrc:{*:4}           // number of SSRCs

[ICE Candidates]
```

As already noted, the Offer/Answer exchange above remains essentially the same regardless of whether the following media session is going to contain one or multiple RTP streams (SSRCs) per medium in either direction.

The exchange also has the following important characteristics:

- o Preserves interoperability with most kinds of legacy endpoints.
- o Allows the negotiation of most parameters that concern the media/RTP stack (typically the browser).
- o Only a single Offer/Answer exchange is required for session establishment and, in most cases, for the entire duration of a session.
- o Leaves complete freedom to applications as to the way that they are going to signal any other information such as SSRC identification information or the addition or removal of RTP streams.

2.1. Discovery

It is important that an implementation using "m=" lines as an envelope for multiple RTP media streams, be able to reliably detect whether its peer is capable of receiving them. One way of achieving this would be the use of upper-layer protocols as explained in Section 3.

In cases where endpoints need to be able to detect this from the SDP Offer/Answer they could use the "max-send-ssrc" and "max-recv-ssrc" attributes defined in [MAX-SSRC]. It has to be noted however that this mechanism is still a work in progress and as such it would still need to be extended to provide ways of distinguishing between independent flows and complementary ones such as layered FEC and RTX.

Unless an endpoint detects the corresponding max-ssrc or upper level protocol indicators that a remote peer can actually handle multiple streams within a single "m=" line, it MUST assume that such support is unavailable.

2.2. Advantages

The advantages of using "m=" lines to represent multiple media sources in the way they were originally intended by SDP can be roughly summarized to the following:

- o It works. Existing implementations are already successfully using the approach.
- o No Offer/Answer when adding/removing streams. Improves flexibility for applications allowing them to only signal information that they really need to.
- o No added glare risk. Improves scalability and reliability.
- o No need to pre-announce SSRCs. Improves scalability.
- o Allows apps to choose fine-tuned signalling: Custom, XCON, RFC4575, WebRTC JavaScript, CLUE channels, or even Plan A and Plan B.

Combined, the above set of characteristics allow for a multi-stream management method that gives scalability, flexibility and reliability.

3. Additional Session Control and Signalling

- o Adding and removing RTP streams to an existing session.
- o Accepting and refusing some of them.
- o Identifying SSRCs and obtaining additional metadata for them (e.g. the user corresponding to a specific SSRC).
- o Requesting that additional SSRCs be added.
- o Requesting that specific processing be applied on some SSRCs.

Support for any subset of the above semantics is highly dependent on the use cases and applications where they are employed. The position of this specification is that they should therefore be left to protocols that target more specific scenarios. There are numerous existing or emerging solutions, some of them developed by the IETF, that already cover this. This includes CLUE channels [CLUE], the SIP Event Package For Conference State [RFC4575] and its XMPP variant [COIN], as well as the protocols defined within the Centralised Conferencing IETF working group [XCON]. Additional mechanisms are very likely to emerge in the future as various applications address

specific use cases, scenarios and topologies. Examples for this could be WebRTC JavaScript applications using proprietary JSON descriptors, XMPP clients with new XML schemas and many others.

The most important part of this specification is hence to prevent certain assumptions or topologies from being imposed on applications. One example of this is the need to know and include in the Offer/Answer exchange, all the SSRCs that can show up in a session. This can be particularly problematic for scenarios that involve endpoints with varying constraints.

Large scale conference calls, potentially federated through RTP translator-like bridges, would be another problematic scenario. Being able to always pre-announce SSRCs in such situations could of course be made to work but it would come at a price. It would either require a very high number of Offer/Answer updates that propagate the information through the entire topology, or use of tricks such as pre-allocating a range of "fake" SSRCs, announcing them to participants and then overwriting the actual SSRCs with them. Depending on the scenario both options could prove inappropriate or inefficient while some applications may not even need such information. Others could be retrieving it through simplistic means such as access to a centralized resource (e.g. an URL pointing to a JSON description of the conference).

4. Demultiplexing and Identifying Streams (Use of Bundle)

For reasons of optimising traversal of Network Address Translation (NAT) gateways, it is likely for endpoints to use [BUNDLE]. This implies that all RTP streams would in many cases end up being received on the same port. A demuxing mechanism is therefore necessary in order for these packets to then be fed into the appropriate processing chain (i.e. matched to an "m=" line).

Note: it is important to distinguish between the demultiplexing and the identification of incoming flows. Throughout this specification the former is used to refer to the process of choosing selecting a depacketizing/decoding/processing chain to feed incoming packets to. Such decisions depend solely on the format that is used to encode the content of incoming packets.

The above is not to be confused with the process of making rendering decision about a processed flow. Such decisions include showing a "current speaker" flow at a specific location, window or video tag, while choosing a different one for a second, "slides" flow. Another example would be the possibility to attach "Alice", "Bob" and "Carol" labels on top of the appropriate UI components. This specification leaves such rendering choices entirely to application-specific signalling as described in Section 3.

This specification uses demuxing based on RTP payload types. When creating offers and answers applications MUST therefore allocate RTP payload types only once per bundle group. In cases where rtcp-mux is in use this would mean a maximum of 96 payload types per bundle [RFC5761]. It has been pointed out that some legacy devices may have unpredictable behaviour with payload types that are outside the 96-127 range reserved by [RFC3551] for dynamic use. Some applications or implementations may therefore choose not to use values outside this range. Whatever the reason, offerers that find they need more than the available payload type numbers, will simply need to either use a second bundle group or not use BUNDLE at all (which in the case of a single audio and a single video "m=" line amounts to roughly the same thing). This would also imply building a dynamic table, mapping SSRCs to PTs and m= lines, in order to then also allow for RTCP demuxing.

While not desirable, the implications of such a decision would be relatively limited. Use of trickle ICE [TRICKLE-ICE] is going to lessen the impact on call establishment latency. Also, the fact that this would only occur in a limited number of cases makes it unlikely to have a significant effect on port consumption.

An additional requirement that has been expressed toward demuxing is the ability to assign incoming packets with the same payload type to different processing chains depending on their SSRCs. A possible example for this is a scenario where two video streams are being rendered on different video screens that each have their own decoding hardware.

While the above may appear as a demuxing and a decoding related problem it is really mostly a rendering policy specific to an application. As such it should be handled by app. specific signalling that could involve custom-formatted, per-SSRC information that accompanies SDP offers and answers.

5. Simulcasting, FEC, Layering and RTX (Open Issue)

Repair flows such as layering, FEC, RTX and to some extent simulcasting, present an interesting challenge, which is why they are considered an open issue by this specification.

On the one hand they are transport utilities that need to be understood, supported and used by the level of the media libraries in a way that is mostly transparent to applications. On the other, some applications may need to be made aware of them and given the option to control their use. This could be necessary in cases where their use needs to be signalled to endpoints through application-specific non-SDP mechanisms. Another example is the possibility for an application to choose to disable some or all repair flows because it has been made aware by application-specific signalling that they are temporarily not being used/rendered by the remote end (e.g. because it is only displaying a thumbnail or because a corresponding video tag is not currently visible).

One way of handling such flows would be to advertise them in the way suggested by [RFC5956] and to then control them through application specific signalling. This options has the merit of already existing but it also implies the pre-announcement and propagation of SSRCs and the bloated signalling that this incurs. Also, relying solely on Offer/Answer here would expose an offerer to the typical race condition of repair SSRCs arriving before the answer and the processing ambiguity that this would imply.

Another approach could be a combination of RTCP and RTP header extensions [RFC5285] in a way similar to the one employed by the Rapid Synchronisation of RTP Flows [RFC6051]. While such a mechanism is not currently defined by the IETF, specifying it could be relatively straightforward:

Every packet belonging to a repair flow could carry an RTP header extension [RFC5285] that points to the source stream (or source layer in case of layered mechanisms).

Again, these are just some possibilities. Different mechanisms may and probably will require different extensions or signalling ([SRCNAME] will likely be an option for some). In some cases, where layering information is provided by the codec, an extensions is not going to be necessary at all.

In cases where FEC or simulcast relations are not immediately needed by the recipient, this information could also be delayed until the reception of the first RTCP packet.

6. Problems with Plans A and B

As already mentioned both Plans A and B heavily rely on SDP and Offer/Answer for advanced stream control. They both require Offer/Answer exchanges in a number of situations where this could be avoided, particularly when adding or removing media sources to a session. This requirement applies equally to cases where a client adds the stream of a newly activated web cam, a simulcast flow or upon the arrival or departure of a conference participant.

Plan A handles such notifications with the addition or removal of independent m= lines [PlanA], while Plan B relies on the use of multiplexed m= lines but still depends on the Offer/Answer exchanges for the addition or removal of media stream identifiers [MSID].

By taking the Offer/Answer approach, both Plan A and Plan B take away from the application the opportunity to handle such events in a way that is most fitting for the use case, which, among other things, also goes against the working group's decision to not to define a specific signalling protocol. (It could be argued that it is therefore only natural how proponents of each plan, having different use cases in mind, are remarkably far from reaching consensus).

Reliance on preliminary announcement of SSRC identifiers is another issue. While this could be perceived as relatively straightforward in one-to-one sessions or even conference calls within controlled environments, it can be a problem in the following cases:

- o interoperability with legacy endpoints
- o use within non-controlled and potentially federated conference environments where new RTP streams may appear relatively often. In such cases the signalling required to describe all of them through Offer/Answer may represent substantial overhead while none or only a part of it (e.g. the description of a main, active speaker stream) may be required by the application.

By increasing the number of Offer/Answer exchanges Both Plan A and Plan B also increase the risk of encountering glare situations (i.e. cases where both parties attempt to modify a session at the same time). While glare is also possible with basic Offer/Answer and resolution of such situations must be implemented anyway, the need to frequently resort to such code may either negatively impact user experience (e.g. when "back off" resolution is used) or require substantial modifications in the Offer/Answer model and/or further venturing into the land of signalling protocols [ROACH-GLARELESS-ADD].

7. IANA Considerations

None.

8. Informative References

- [BUNDLE] Holmberg, C., Alvestrand, H., and C. Jennings, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", reference.I-D.ietf-clue-framework (work in progress), June 2013, <reference.I-D.ietf-mmusic-sdp-bundle-negotiation>.
- [CLUE] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", reference.I-D.ietf-clue-framework (work in progress), May 2013, <reference.I-D.ietf-clue-framework>.
- [COIN] Ivov, E. and E. Marocco, "XEP-0298: Delivering Conference Information to Jingle Participants (Coin)", XSF XEP 0298, June 2011, <reference.I-D.ietf-coin-framework>.
- [MAX-SSRC] Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization sources (SSRC) in RTP Session Signaling ", reference.I-D.westerlund-avtcore-max-ssrc (work in progress), July 2012, <reference.I-D.westerlund-avtcore-max-ssrc>.
- [MSID] Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol", reference.I-D.ietf-mmusic-msid (work in progress), February 2013, <reference.I-D.ietf-mmusic-msid>.
- [PlanA] Roach, A. and M. Thomson, "Using SDP with Large Numbers of Media Flows", reference.I-D.roach-rtcweb-plan-a (work in progress), May 2013, <reference.I-D.roach-rtcweb-plan-a>.
- [PlanB] Uberti, J., "Plan B: a proposal for signaling multiple media sources in WebRTC.", reference.I-D.uberti-rtcweb-plan (work in progress), May 2013, <reference.I-D.uberti-rtcweb-plan>.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [ROACH-GLARELESS-ADD]
Roach, A., "An Approach for Adding RTCWEB Media Streams without Glare", reference.I-D.roach-rtcweb-glareless-add (work in progress), May 2013, <reference.I-D.roach-rtcweb-glareless-add>.
- [SRCNAME] Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDES Item SRCNAME to Label Individual Sources ", reference.I-D.westerlund-avtext-rtcp-sdes-srcname (work in progress), October 2012, <reference.I-D.westerlund-avtext-rtcp-sdes-srcname>.
- [TRICKLE-ICE]
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol ", reference.I-D.ivov-mmusic-trickle-ice (work in progress), March 2013, <reference.I-D.ivov-mmusic-trickle-ice>.
- [XCON] , "Centralized Conferencing (XCON) Status Pages", , <<http://tools.ietf.org/wg/xcon/>>.

Appendix A. Acknowledgements

Many thanks to Jonathan Lennox for reviewing the document and providing valuable feedback.

Author's Address

Emil Ivov
Jitsi
Strasbourg 67000
France

Phone: +33-177-624-330
Email: emcho@jitsi.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2013

E. Iovov
Jitsi
E.K. Rescorla
RTFM, Inc.
J. Uberti
Google
March 11, 2013

Trickle ICE: Incremental Provisioning of Candidates for the Interactive
Connectivity Establishment (ICE) Protocol
draft-iovov-mmusic-trickle-ice-01

Abstract

This document describes an extension to the Interactive Connectivity Establishment (ICE) protocol that allows ICE agents to send and receive candidates incrementally rather than exchanging complete lists. With such incremental provisioning, ICE agents can begin connectivity checks while they are still gathering candidates and considerably shorten the time necessary for ICE processing to complete.

The above mechanism is also referred to as "trickle ICE".

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 September 12, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Incompatibility with Standard ICE	5
4. Determining Support for Trickle ICE	6
4.1. Unilateral Use of Trickle ICE (Half Trickle)	7
5. Sending the Initial Offer	8
5.1. Encoding the SDP	9
6. Receiving the Initial Offer	9
6.1. Sending the Initial Answer	10
6.2. Forming check lists and beginning connectivity checks	10
6.3. Encoding the SDP	11
7. Receiving the Initial Answer	11
8. Performing Connectivity Checks	11
8.1. Check List and Timer State Updates	11
9. Discovering and Sending Additional Local Candidates	12
9.1. Pairing newly learned candidates and updating check lists	12
9.2. Encoding the SDP for Additional Candidates	14
9.3. Announcing End of Candidates	14
9.4. Receiving an End Of Candidates Notification	16
10. Receiving Additional Remote Candidates	16
11. Concluding ICE Processing	16
12. Subsequent Offer/Answer Exchanges	17
13. Interaction with ICE Lite	17
14. Example Flow	18
15. Security Considerations	19
16. Acknowledgements	19
17. References	19
17.1. Normative References	19
17.2. Informative References	19
Appendix A. Open issues	21
A.1. MID/Stream Indices in SDP	21
A.2. Starting checks	21
Appendix B. Changes From Earlier Versions	21
B.1. Changes From draft-ivov-00	21

B.2. Changes From draft-rescorla-01	22
B.3. Changes From draft-rescorla-00	23
Authors' Addresses	23

1. Introduction

The Interactive Connectivity Establishment (ICE) protocol [RFC5245] describes mechanisms for gathering, candidates, prioritizing them, choosing default ones, exchanging them with the remote party, pairing them and ordering them into check lists. Once all of the above have been completed, and only then, the participating agents can begin a phase of connectivity checks and eventually select the pair of candidates that will be used in the following session.

While the above sequence has the advantage of being relatively straightforward to implement and debug once deployed, it may also prove to be rather lengthy. Gathering candidates or candidate harvesting would often involve things like querying STUN [RFC5389] servers, discovering UPnP devices, and allocating relayed candidates at TURN [RFC5766] servers. All of these can be delayed for a noticeable amount of time and while they can be run in parallel, they still need to respect the pacing requirements from [RFC5245], which is likely to delay them even further. Some or all of the above would also have to be completed by the remote agent. Both agents would next perform connectivity checks and only then would they be ready to begin streaming media.

All of the above could lead to relatively lengthy session establishment times and degraded user experience.

The purpose of this document is to define an alternative mode of operation for ICE implementations, also known as "trickle ICE", where candidates can be exchanged incrementally. This would allow ICE agents to exchange host candidates as soon as a session has been initiated. Connectivity checks for a media stream would also start as soon as the first candidates for that stream have become available.

Trickle ICE allows reducing session establishment times in cases where connectivity is confirmed for the first exchanged candidates (e.g. where the host candidates for one of the agents are directly reachable from the second agent). Even when this is not the case, running candidate harvesting for both agents and connectivity checks all in parallel allows to considerably reduce ICE processing times.

It is worth pointing out that before being introduced to the IETF, trickle ICE had already been included in specifications such as XMPP Jingle [XEP-0176] and it has been in use in various implementations and deployments.

In addition to the basics of trickle ICE, this document also describes how support for trickle ICE needs to be discovered, how regular ICE processing needs to be modified when building and updating check lists, and how trickle ICE implementations should interoperate with agents that only implement [RFC5245] processing.

This specification does not define usage of trickle ICE with any specific signalling protocol, contrary to [RFC5245] which contains a usage for ICE with SIP. Such usages would have to be specified in separate documents such as for example [I-D.ivov-mmusic-trickle-ice-sip].

Trickle ICE does however reuse and build upon the SDP syntax defined by vanilla ICE.

2. Terminology

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

This specification makes use of all terminology defined by the protocol for Interactive Connectivity Establishment in [RFC5245].

Vanilla ICE: The Interactive Connectivity Establishment protocol as defined in [RFC5245].

Candidate Harvester: A module used by an ICE agent to obtain local candidates. Candidate harvesters use different mechanisms for discovering local candidates. Some of them would typically make use of protocols such as STUN or TURN. Others may also employ techniques that are not referenced within [RFC5245]. UPnP based port allocation and XMPP Jingle Relay Nodes [XEP-0278] are among the possible examples.

Trickled Candidates: Candidates that a trickle ICE agent is sending subsequently to but within the context defined by an offer or an answer. Trickled candidates can be sent in parallel with candidate harvesting and connectivity checks.

Trickling/Trickle (v.): The act of sending trickled candidates.

Half Trickle: A trickle ICE mode of operation where the offerer gathers its first generation of candidates strictly before creating and sending the offer. Once sent, that offer can be processed by vanilla ICE agents and does not require support for this specification. It also allows trickle ICE capable answerers to still gather candidates and perform connectivity checks in a non-blocking way, thus roughly offering "half" the advantages of trickle ICE. The mechanism is mostly meant for use in cases where support for trickle ICE cannot be confirmed prior to sending a first offer.

Full Trickle: Regular mode of operation for trickle ICE agents, used in opposition to the half trickle mode of operation.

3. Incompatibility with Standard ICE

The ICE protocol was designed to be fairly flexible so that it would work in and adapt to as many network environments as possible. It is hence important to point out at least some of the reasons why, despite its flexibility, the specification in [RFC5245] would not support trickle ICE.

[RFC5245] describes the conditions required to update check lists and timer states while an ICE agent is in the Running state. These conditions are verified upon transaction completion and one of them stipulates that:

If there is not a pair in the valid list for each component of the media stream, the state of the check list is set to Failed.

This could be a problem and cause ICE processing to fail prematurely in a number of scenarios. Consider the following case:

- o Alice and Bob are both located in different networks with Network Address Translation (NAT). Alice and Bob themselves have different address but both networks use the same [RFC1918] block.
- o Alice sends Bob the candidate 10.0.0.10 which also happens to correspond to an existing host on Bob's network.
- o Bob creates a check list consisting solely of 10.0.0.10 and starts checks.
- o These checks reach the host at 10.0.0.10 in Bob's network, which responds with an ICMP "port unreachable" error and per [RFC5245] Bob marks the transaction as Failed.

At this point the check list only contains Failed candidates and the valid list is empty. This causes the media stream and potentially all ICE processing to Fail.

A similar race condition would occur if the initial offer from Alice only contains candidates that can be determined as unreachable (per [I-D.keranen-mmusic-ice-address-selection]) from any of the candidates that Bob has gathered. This would be the case if Bob's candidates only contain IPv4 addresses and the first candidate that he receives from Alice is an IPv6 one.

Another potential problem could arise when a non-trickle ICE implementation sends an offer to a trickle one. Consider the following case:

- o Alice's client has a non-trickle ICE implementation
- o Bob's client has support for trickle ICE.
- o Alice and Bob are behind NATs with address-dependent filtering [RFC4787].
- o Bob has two STUN servers but one of them is currently unreachable

After Bob's agent receives Alice's offer it would immediately start connectivity checks. It would also start gathering candidates, which would take long because of the unreachable STUN server. By the time Bob's answer is ready and sent to Alice, Bob's connectivity checks may well have failed: until Alice gets Bob's answer, she won't be able to start connectivity checks and punch holes in her NAT. The NAT would hence be filtering Bob's checks as originating from an unknown endpoint.

4. Determining Support for Trickle ICE

According to [RFC5245] every time an agent supporting trickle ICE generates an offer or an answer, it MUST include the "trickle" token in the ice-options attribute. Syntax for this token is defined in Section 5.1.

Additionally, in order to avoid interoperability problems such as those described in Section 3, it is important that trickle ICE negotiation is only attempted in cases where the remote party actually supports this specification. Agents that receive offers or answers can verify support by examining them for the "trickle" ice-options token. However, agents that are about to send a first offer, have no immediate way of doing this. This means that usages of trickle for specific protocols would need to either:

- o Provide a way for agents to verify support of trickle ICE prior to initiating a session. XMPP's Service discovery [XEP-0030] is an example for one such mechanism;
- o Make support for trickle ICE mandatory so that support could be assumed the agents.

Alternately, for cases where a protocol provides neither of the above, agents may either rely on provisioning/configuration, or use the half trickle procedure described in Section 4.1.

Note that out-of-band discovery semantics and half trickle are only necessary prior to session initiation, or in other words, when sending the initial offer. Once a session is established and trickle ICE support is confirmed for both parties, either agent can use full trickle for subsequent offers.

4.1. Unilateral Use of Trickle ICE (Half Trickle)

The idea of using half trickle is about having the caller send a regular, vanilla ICE offer, with a complete set of candidates. This offer still indicates support for trickle ice, so the answerer is able to respond with an incomplete set of candidates and continue trickling the rest. Half trickle offers will typically contain an end-of-candidates indication.

The mechanism can be used in cases where there is no way for an agent to verify in advance whether a remote party supports trickle ice. Because it contains a full set of candidates, its first offer can thus be handled by a regular vanilla ICE agent, while still allowing a trickle one to use the optimisation defined in this specification. This prevents negotiation from failing in the former case while still giving roughly half the trickle ICE benefits in the latter (hence the name of the mechanism).

Use of half trickle is only necessary during an initial offer/answer exchange. Once both parties have received a session description from their peer, they can each reliably determine trickle ICE support and use it for all subsequent offer/answer exchanges.

It is worth pointing out that using half trickle may actually bring more than just half the improvement in terms of user experience. This can happen in cases where an agent starts gathering candidates upon user interface cues that a call is pending, such as activity on a keypad or the phone going off hook. This would mean a part or all candidate harvesting could have completed before the agent actually needs to send the offer. Given that the answerer will be able to trickle candidates, both agents will be able to start connectivity

checks and complete ICE processing earlier than with vanilla ICE and potentially even as early as with full trickle.

However, such anticipation is not not always possible. For example, a multipurpose user agent or a WebRTC web page where communication is a non-central feature (e.g. calling a support line in case of a problem with the main features) would not necessarily have a way of distinguishing between call intentions and other user activity. Still, even in these cases, using half trickle would be an improvement over vanilla ICE as it would optimize performance for answerers.

5. Sending the Initial Offer

An agent starts gathering candidates as soon as it has an indication that communication is imminent (e.g. a user interface cue or an explicit request to initiate a session). Contrary to vanilla ICE, implementations of trickle ICE do not need to gather candidates in a blocking manner. Therefore, unless half trickle is being used, agents SHOULD generate and transmit their initial offer as early as possible, in order to allow the remote party to start gathering and trickling candidates.

Trickle ICE agents MAY include any set of candidates in an offer. This includes the possibility of generating one with no candidates, or one that contains all the candidates that the agent is planning on using in the following session.

For optimal performance, it is RECOMMENDED that an initial offer contains host candidates only. This would allow both agents to start gathering server reflexive, relayed and other non-host candidates simultaneously, and it would also enable them to begin connectivity checks.

If the privacy implications of revealing host addresses are a concern, agents MAY generate an offer that contains no candidates and then only trickle candidates that do not reveal host addresses (e.g. relayed candidates).

Prior to actually sending an initial offer, agents MAY verify if the remote party supports trickle ICE, where such mechanisms actually exist. If absence of such support is confirmed agents MUST fall back to using vanilla ICE or abandon the entire session.

All trickle ICE offers and answers MUST indicate support of this specification, as explained in Section 5.1.

Calculating priorities and foundations, as well as determining redundancy of candidates work the same way they do with vanilla ICE.

5.1. Encoding the SDP

The process of encoding the SDP [RFC4566] is mostly the same as the one used by vanilla ICE. Still, trickle ICE does require a few differences described here.

Agents MUST indicate support for Trickle ICE by including the "trickle" token for the "a=ice-options" attribute:

```
a=ice-options:trickle
```

As mentioned earlier in this section, Offers and Answers can contain any set of candidates, which means that a trickle ICE session description MAY contain no candidates at all. In such cases the agent would still need to place an address in the "c=" line(s). If the use of a host address there is undesirable (e.g. for privacy reasons), the agent MAY set the connection address to IP6 ::. In this case it MUST also set the port number to 9 (Discard). There is no need to include a fictitious candidate for the IP6 :: address when doing so.

It is worth noting that the use of IP6 :: has been selected over IP4 0.0.0.0, even though [RFC3264] already gives the latter semantics appropriate for such use. The reason for this choice is the historic use of 0.0.0.0 as a means of putting a stream on hold [RFC2543] and the ambiguity that this may cause with legacy libraries and applications.

It is also worth mentioning that use of IP6 :: here does not constitute any kind of indication as to the actual use of IPv6 candidates in a session and it can very well appear in a negotiation that only involves IPv4 candidates.

6. Receiving the Initial Offer

When an agent receives an initial offer, it will first check if it indicates support for trickle ICE as explained in Section 4. If this is not the case, the agent MUST process the offer according to the [RFC5245] procedures or standard [RFC3264] processing in case no ICE support is detected at all.

It is worth pointing out that in case support for trickle ICE is confirmed, an agent will automatically assume support for vanilla ICE as well even if the support verification procedure in [RFC5245] indicates otherwise. Specifically, such verification would indicate lack of support when the offer contains no candidates. The IP6 :: address present in the c= line in that case would not "appear in a candidate attribute". Obviously, a fallback to [RFC3264] is not required when this happens.

If, the offer does indicate support for trickle ICE, the agent will determine its role, start gathering and prioritizing candidates and, while doing so it will also respond by sending its own answer, so that both agents can start forming check lists and begin connectivity checks.

6.1. Sending the Initial Answer

An agent can respond to an initial offer at any point while gathering candidates. The answer can again contain any set of candidates including none or all of them. Unless it is protecting host addresses for privacy reasons, the agent would typically construct this initial answer including only them, thus allowing the remote party to also start forming checklists and performing connectivity checks.

The answer MUST indicate support for trickle ICE as described by Section 4.

6.2. Forming check lists and beginning connectivity checks

After exchanging offer and answer, and as soon as they have obtained local and remote candidates, agents will begin forming candidate pairs, computing their priorities and creating check lists according to the vanilla ICE procedures described in [RFC5245]. Obviously in order for candidate pairing to be possible, it would be necessary that both the offer and the answer contained candidates. If this was not the case agents will still create the check lists (so that their Active/Frozen state could be monitored and updated) but they will only populate them once they actually have the candidates.

Initially, all check lists will have their Active/Frozen state set to Frozen.

Trickle ICE agents will then also attempt to unfreeze the check list for the first media stream (i.e. the first media stream that was reported to the ICE implementation from the using application). If this checklist is still empty however, agents will continue examining media streams in the order they were reported and will unfreeze the first non-empty checklist.

Respecting the order in which lists have been reported to an ICE implementation, or in other words, the order in which they appear in SDP, is helpful so that checks for the same media stream is more likely to be performed simultaneously by both agents.

6.3. Encoding the SDP

The process for encoding the SDP at the answerer is identical to the process followed by the offerer for both full and lite implementations, as described in Section 5.1.

7. Receiving the Initial Answer

When receiving an answer, agents will follow vanilla ICE procedures to determine their role and they would then form check lists (as described in Section 6.2) and begin connectivity checks .

8. Performing Connectivity Checks

For the most part, trickle ICE agents perform connectivity checks following vanilla ICE procedures. Of course, the asynchronous nature of candidate harvesting in trickle ICE would impose a number of changes described here.

8.1. Check List and Timer State Updates

The vanilla ICE specification requires that agents update check lists and timer states upon completing a connectivity check transaction. During such an update vanilla ICE agents would set the state of a check list to Failed if the following two conditions are satisfied:

- o all of the pairs in the check list are either in the Failed or Succeeded state;
- o if at least one of the components of the media stream has no pairs in its valid list.

With trickle ICE, the above situation would often occur when candidate harvesting and trickling are still in progress and it is perfectly possible that future checks will succeed. For this reason trickle ICE agents add the following conditions to the above list:

- o all candidate harvesters have completed and the agent is not expecting to learn any new candidates;
- o the remote agent has sent an end-of-candidates indication for that check list as described in Section 9.3.

Vanilla ICE requires that agents then update all other check lists, placing one pair in each of them into the Waiting state, effectively unfreezing the check list. Given that with trickle ICE, other check lists may still be empty at that point, a trickle ICE agent SHOULD also maintain an explicit Active/Frozen state for every check list, rather than deducing it from the state of the pairs it contains. This state should be set to Active when unfreezing the first pair in a list or when that couldn't happen because a list was empty.

9. Discovering and Sending Additional Local Candidates

After an offer or an answer have been sent, agents will most likely continue discovering new local candidates as STUN, TURN and other non-host candidate harvesting mechanisms begin to yield results. Whenever an agent discovers such a new candidate it will compute its priority, type, foundation and component id according to normal vanilla ICE procedures.

The new candidate is then checked for redundancy against the existing list of local candidates. If its transport address and base match those of an existing candidate, it will be considered redundant and will be ignored. This would often happen for server reflexive candidates that match the host addresses they were obtained from (e.g. when the latter are public IPv4 addresses). Contrary to vanilla ICE, trickle ICE agents will consider the new candidate redundant regardless of its priority. [TODO: is this OK? if not we need to check if the existing candidate was already used in conn checks, cancel them, and then restart them with the new candidate ... and in this specific case there's probably no point to do that].

Then, if no remote candidates are currently known for this same stream, the new candidate will simply be added to the list of local candidates.

Otherwise, if the agent has already learned of one or more remote candidates for this stream and component, it will begin pairing the new local candidates with them and adding the pairs to the existing check lists according to their priority.

9.1. Pairing newly learned candidates and updating check lists

Forming candidate pairs will work the way it is described by the vanilla ICE specification. Actually adding the new pair to a check list however, will happen according to the rules described below.

If the new pair's local candidate is server reflexive, the server reflexive candidate **MUST** be replaced by its base before adding the pair to the list. Once this is done, the agent examines the check list looking for another pair that would be redundant with the new one. If such a pair exists and its state is:

Succeeded: the newly formed pair is ignored.

Frozen or Waiting: the agent chooses the pair with the higher priority local candidate, places it in the state that the old pair was in (i.e. Frozen or Waiting) and removes the other one as redundant.

Failed: the agent chooses the pair with the higher priority local candidate, places it in the Waiting state and removes the other one as redundant.

In-Progress: The agent cancels the in-progress transaction (where cancellation happens as explained in Section 7.2.1.4 of [RFC5245]), then it chooses the pair with the higher priority local candidate, places it in the Waiting state and removes the other one as redundant.

For all other pairs, including those with a server reflexive local candidate that were not found to be redundant:

- o if all check lists are empty and in the Frozen state, or in other words, if this is the first pair the agent is adding to any check list, both the pair and its containing check list will be placed in an Active state.
- o if this check list is Frozen then the new pair will also be assigned a Frozen state.
- o else if the check list is Active and it is either empty or contains only candidates in the Succeeded and Failed states, then the new pair's state is set to Waiting.
- o else if the check list is non-empty and Active, then the new pair state will be set to

Frozen: if there is at least one pair in the list whose foundation matches the one in the new pair and whose state is neither Succeeded nor Failed (eventually the new pair

will get unfrozen after the the on-going check for the existing pair concludes);

Waiting: if the list contains no pairs with the same foundation as the new one, or, in case such pairs exist but they are all in either the Succeeded or Failed states.

9.2. Encoding the SDP for Additional Candidates

To facilitate interoperability an ICE agent will encode additional candidates using the vanilla ICE SDP syntax. For example:

```
a=candidate:2 1 UDP 1658497328 198.51.100.33 5000 typ host
```

Given that such lines do not provide a relationship between the candidate and the m line that it relates to, signalling protocols using trickle ICE MUST establish that relation themselves using an MID [RFC3388]. Such MIDs use "media stream identification", as defined in [RFC3388], to identify a corresponding m-line. When creating candidate lines usages of trickle ICE MUST use the MID if possible, or the m-line index if not. Obviously, agents MUST NOT send individual candidates prior to generating the corresponding SDP session description.

The exact means of transporting additional candidates to a remote agent is left to the protocols using trickle ICE. It is important to note, however, that these candidate exchanges are not part of the offer/answer model.

9.3. Announcing End of Candidates

Once all candidate harvesters for a specific media stream complete, or expire, the agents will generate an "end-of-candidates" indication for that stream and send it to the remote agent via the signalling channel. Such indications are sent in the form of a media-level attribute that has the following form: end-of-candidates.

```
a=end-of-candidates
```

The end-of-candidates indications can be sent as part of an offer, which would typically be the case with half trickle initial offers, they can accompany the last candidate an agent can send for a stream,

and they can also be sent alone (e.g. after STUN Binding requests or TURN Allocate requests to a server timeout and the agent has no other active harvesters).

Controlled trickle ICE agents SHOULD always send end-of-candidates indications once harvesting for a media stream has completed unless ICE processing terminates before they've had a chance to do so. Sending the indication is necessary in order to avoid ambiguities and speed up ICE conclusion. This is necessary in order to avoid ambiguities and speed up ICE conclusion. Controlling agents on the other hand MAY sometimes conclude ICE processing prior to sending end-of-candidates notifications for all streams. This would typically be the case with aggressive nomination. Yet it is RECOMMENDED that controlling agents do send such indications whenever possible for the sake of consistency and keeping middle boxes and controlled agents up-to-date on the state of ICE processing.

When sending end-of-candidates during trickling, rather than as a part of an offer or an answer, it is the responsibility of the using protocol to define means that can be used to relate the indication to one or more specific m-lines.

Receiving an end-of-candidates notification allows an agent to update check list states and, in case valid pairs do not exist for every component in every media stream, determine that ICE processing has failed. It also allows agents to speed ICE conclusion in cases where a candidate pair has been validated but it involves the use of lower-preference transports such as TURN. In such situations some implementations may choose to wait in case higher-priority candidates are received and end-of-candidates provides an indication that this is not going to happen.

An agent MAY also choose to generate an end-of-candidates event before candidate harvesting has actually completed, if the agent determines that harvesting has continued for more than an acceptable period of time. However, an agent MUST NOT send any more candidates after it has send an end-of-candidates notification.

When performing half trickle agents SHOULD send end-of-candidates together with their initial offer unless they are planning on potentially sending additional candidates in case the remote party turns out to actually support trickle ICE.

When end-of-candidates is sent as part of an offer or an answer it can appear as a session-level attribute, which would be equivalent to having it appear in all m-lines.

Once an agent sends the end-of-candidates event, it will update the state of the corresponding check list as explained in section Section 8.1. Past that point agents MUST NOT send any new candidates. Once an agent has received an end-of-candidates indication, it MUST also ignore any newly received candidates for that media stream. Adding new candidates to the negotiation is hence only possible through an ICE restart.

It is important to note that This specification does not override vanilla ICE semantics for concluding ICE processing. This means that even if end-of-candidates indications are sent agents will still have to go through pair nomination. Also, if pairs have been nominated for components and media streams, ICE processing will still conclude even if end-of-candidate indications have not been received for all streams.

9.4. Receiving an End Of Candidates Notification

When an agent receives an end-of-candidates notification for a specific check list, they will update its state as per Section 8.1. In case the list is still in the Active state after the update, the agent will persist the the fact that an end-of-candidates notification has been received for and take it into account in future list updates.

[TODO would we like to say anything about nomination? in general this would be up to implementers but is there a need for some basic guidelines?]

10. Receiving Additional Remote Candidates

At any point of ICE processing, a trickle ICE agent may receive new candidates from the remote agent. When this happens and no local candidates are currently known for this same stream, the new remote candidates are simply added to the list of remote candidates.

Otherwise, the new candidates are used for forming candidate pairs with the pool of local candidates and they are added to the local check lists as described in Section 9.1.

Once the remote agent has completed candidate harvesting, it will send an end-of-candidates event. Upon receiving such an event, the local agent MUST update check list states as per Section 8.1. This may lead to some check lists being marked as Failed.

11. Concluding ICE Processing

This specification does not directly modify the procedures ending ICE processing described in Section 8 of [RFC5245], and trickle ICE implementations will follow the same rules.

12. Subsequent Offer/Answer Exchanges

Either agent MAY generate a subsequent offer at any time allowed by [RFC3264]. When this happens agents will use [RFC5245] semantics to determine whether or not the new offer requires an ICE restart. If this is the case then agents would perform trickle ICE as they would in an initial offer/answer exchange.

The only differences between an ICE restart and a brand new media session are that:

- o during the restart, media can continue to be sent to the previously validated pair.
- o both agents are already aware whether or not their peer supports trickle ICE, and there is no longer need for performing half trickle or confirming support with other mechanisms.

13. Interaction with ICE Lite

Behaviour of Trickle ICE capable ICE lite agents does not require any particular rules other than those already defined in this specification and [RFC5245]. This section is hence added with an informational purpose only.

A Trickle ICE capable ICE Lite agent would generate offers or answers as per [RFC5245]. Both will indicate support for trickle ICE (Section 5.1) and given that they will contain a complete set of candidates (the agent's host candidates) these offers and answers would also be accompanied with an end-of-candidates notification.

When performing full trickle, a full ICE implementation could send an offer or an answer with no candidates and an `IP6 :: connection line address`. After receiving an answer that identifies the remote agent as an ICE lite implementation, the offerer may very well choose to not send any additional candidates. The same is also true in the case when the ICE lite agent is making the offer and the full ICE one is answering. In these cases the connectivity checks would be enough for the ICE lite implementation to discover all potentially useful candidates as peer reflexive. The following example illustrates one such ICE session:

ICE Lite

Bob

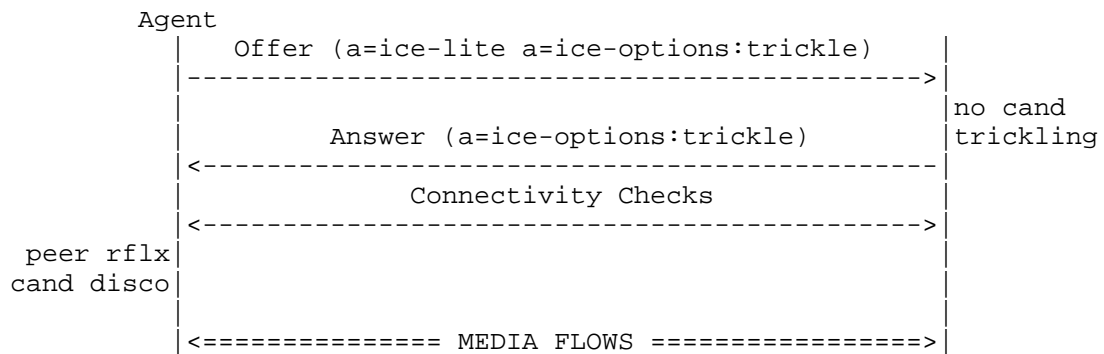


Figure 1: Example

In addition to reducing signaling traffic this approach also removes the need to discover STUN bindings, or to make TURN or UPnP allocations which may considerably lighten ICE processing.

14. Example Flow

A typical successful trickle ICE exchange with an Offer/Answer protocol would look this way:

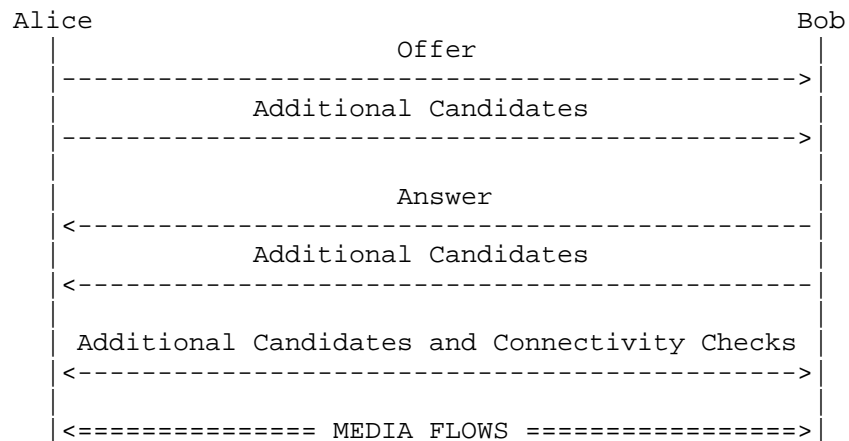


Figure 2: Example

15. Security Considerations

[TODO]

16. Acknowledgements

The authors would like to thank Bernard Adoba, Christer Holmberg, Enrico Marocco, Flemming Andreassen, Jonathan Lennox and Martin Thomson for their reviews and suggestions on improving this document.

17. References

17.1. 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.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

17.2. Informative References

- [I-D.ivov-mmusic-trickle-ice-sip] Ivov, E., Marocco, E., and C. Holmberg, "A Session Initiation Protocol (SIP) usage for Trickle ICE", draft-ivov-mmusic-trickle-ice-sip-00 (work in progress), January 2013.
- [I-D.keranen-mmusic-ice-address-selection] Keraenen, A. and J. Arkko, "Update on Candidate Address Selection for Interactive Connectivity Establishment (ICE)", draft-keranen-mmusic-ice-address-selection-01 (work in progress), July 2012.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

- [RFC2543] Handley, M., Schulzrinne, H., Schooler, E., and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.
- [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.
- [RFC3388] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.
- [RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [XEP-0030] Hildebrand, J., Millard, P., Eatmon, R., and P. Saint-Andre, "XEP-0030: Service Discovery", XEP XEP-0030, June 2008.
- [XEP-0115] Hildebrand, J., Saint-Andre, P., Troncon, R., and J. Konieczny, "XEP-0115: Entity Capabilities", XEP XEP-0115, February 2008.
- [XEP-0176] Beda, J., Ludwig, S., Saint-Andre, P., Hildebrand, J., Egan, S., and R. McQueen, "XEP-0176: Jingle ICE-UDP Transport Method", XEP XEP-0176, June 2009.
- [XEP-0278] Camargo, T., "XEP-0278: Jingle Relay Nodes", XEP XEP-0278, June 2011.

Appendix A. Open issues

At the time of writing of this document the authors have no clear view on how and if the following list of issues should be addressed.

A.1. MID/Stream Indices in SDP

This specification does not currently define syntax for candidate-to-stream bindings although it says that they should be implemented with MID or a stream index. Yet, it is reasonable to assume that most usages would need to do this within the SDP and it may make sense to agree on the format. Here's one possible way to do this:

```
a=mid:1
a=candidate:1 1 UDP 1658497328 192.168.100.33 5000 typ host
a=candidate:2 1 UDP 1658497328 96.1.2.3 5000 typ srflx
a=mid:2
a=candidate:2 1 UDP 1658497328 96.1.2.3 5002 typ srflx
a=end-of-candidates
```

A.2. Starting checks

Normally Vanilla ICE implementations would first activate a check list, validate at least one pair in every component and only then unfreeze all other checklists. With trickle ICE this would be suboptimal since, candidates can arrive randomly and we would be wasting time waiting for a checklist to fill (almost as if we were doing vanilla ICE). We need to decide if unfreezing everything solely based on foundation is good enough.

Appendix B. Changes From Earlier Versions

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

B.1. Changes From draft-ivov-00

- o Specified that end-of-candidates is a media level attribute which can of course appear as session level, which is equivalent to having it appear in all m-lines. Also made end-of-candidates optional for cases such as aggressive nomination for controlled agents.

- o Added an example for ICE lite and trickle ICE to illustrate how, when talking to an ICE lite agent doesn't need to send or even discover any candidates.
- o Added an example for ICE lite and trickle ICE to illustrate how, when talking to an ICE lite agent doesn't need to send or even discover any candidates.
- o Added wording that explicitly states ICE lite agents have to be prepared to receive no candidates over signalling and that they should not freak out if this happens. (Closed the corresponding open issue).
- o It is now mandatory to use MID when trickling candidates and using m-line indexes is no longer allowed.
- o Replaced use of 0.0.0.0 to IP6 :: in order to avoid potential issues with RFC2543 SDP libraries that interpret 0.0.0.0 as an on-hold operation. Also changed the port number here from 1 to 9 since it already has a more appropriate meaning. (Port change suggested by Jonathan Lennox).
- o Closed the Open Issue about use about what to do with cands received after end-of-cands. Solution: ignore, do an ice restart if you want to add something.
- o Added more terminology, including trickling, trickled candidates, half trickle, full trickle,
- o Added a reference to the SIP usage for trickle ICE as requested at the Boston interim.

B.2. Changes From draft-rescorla-01

- o Brought back explicit use of Offer/Answer. There are no more attempts to try to do this in an O/A independent way. Also removed the use of ICE Descriptions.
- o Added SDP specification for trickled candidates, the trickle option and 0.0.0.0 addresses in m-lines, and end-of-candidates.
- o Support and Discovery. Changed that section to be less abstract. As discussed in IETF85, the draft now says implementations and usages need to either determine support in advance and directly use trickle, or do half trickle. Removed suggestion about use of discovery in SIP or about letting implementing protocols do what they want.

- o Defined Half Trickle. Added a section that says how it works. Mentioned that it only needs to happen in the first o/a (not necessary in updates), and added Jonathan's comment about how it could, in some cases, offer more than half the improvement if you can pre-gather part or all of your candidates before the user actually presses the call button.
- o Added a short section about subsequent offer/answer exchanges.
- o Added a short section about interactions with ICE Lite implementations.
- o Added two new entries to the open issues section.

B.3. Changes From draft-rescorla-00

- o Relaxed requirements about verifying support following a discussion on MMUSIC.
- o Introduced ICE descriptions in order to remove ambiguous use of 3264 language and inappropriate references to offers and answers.
- o Removed inappropriate assumption of adoption by RTCWEB pointed out by Martin Thomson.

Authors' Addresses

Emil Ivov
Jitsi
Strasbourg 67000
France

Phone: +33 6 72 81 15 55
Email: emcho@jitsi.org

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Phone: +1 650 678 2350
Email: ekr@rtfm.com

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

Phone: +1 857 288 8888
Email: justin@uberti.name

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 3, 2013

E. Ivov
Jitsi
E. Marocco
Telecom Italia
C. Holmberg
Ericsson
January 30, 2013

A Session Initiation Protocol (SIP) usage for Trickle ICE
draft-ivov-mmusic-trickle-ice-sip-00

Abstract

The Interactive Connectivity Establishment (ICE) protocol describes a Network Address Translator (NAT) traversal for UDP-based multimedia sessions established with the offer/answer model. The ICE extension for Incremental Provisioning of Candidates (Trickle ICE) defines a mechanism that allows ICE agents to shorten session establishment delays by making the candidate gathering and connectivity checking phases of ICE non-blocking.

This document defines usage semantics for Trickle ICE with SIP.

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 August 3, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Half vs Full Trickle	3
4. Encoding and Sending Candidate Information	4
5. Info Package	5
5.1. Overall Description	5
5.2. Applicability	5
5.3. INFO Package Name	5
5.4. INFO Package Parameters	5
5.5. SIP Option-Tags	5
5.6. INFO Message Body Parts	5
6. Example Flows	6
7. Security Considerations	7
8. Acknowledgements	7
9. References	7
9.1. Normative References	7
9.2. Informative References	7
Appendix A. Open issues	8
Authors' Addresses	8

1. Introduction

The vanilla specification of the Interactive Connectivity Establishment (vanilla ICE) protocol [RFC5245] describes a mechanism for NAT traversal that consists of three main phases: a phase where an agent gathers a set of candidate 5-tuples (source IP address and port, destination IP address and port and a transport protocol), a second phase where these candidates are sent to a remote agent and this gathering is repeated and then a third phase where connectivity between all candidates in both sets is checked (connectivity checks). Only then can both agents begin communication, provided of course that ICE processing has successfully completed. According to that specification the three phases above happen consecutively, in a blocking way, which may lead to undesirable latency during session establishment.

The trickle ICE extension defined in [I-D.ivov-mmusic-trickle-ice] defines generic semantics required for these ICE phases to happen simultaneously, in a non-blocking way and hence speed up session establishment.

This specification defines a usage of trickle ICE with the Session Initiation Protocol (SIP). It describes how and when SIP agents use the full and half trickle modes of operation, how they encode additional candidates and how they exchange them through use of SIP INFO requests.

This document also defines a new Info Package for use with Trickle ICE.

2. Terminology

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

This specification makes use of all terminology defined by the protocol for Interactive Connectivity Establishment in [RFC5245] and its Trickle ICE extension [I-D.ivov-mmusic-trickle-ice]. It is assumed that the reader will be familiar with the terminology from both of them.

3. Half vs Full Trickle

Trickle ICE defines a mode of operation called "half trickle". With half trickle the first offer in a session contains all candidates and

subsequent trickling occurs from the offerer in this first offer/answer negotiation. Half trickle offers can hence be processed by both vanilla and trickle ICE agents, which offers an interesting advantage in cases where support for trickle cannot be verified prior to sending an offer.

Unless agents are running within controlled environments or using GRUU, this would be the case with SIP. In spite of mechanisms such as the one defined in [RFC3840], a SIP UA cannot rely on consecutive requests reaching the same destination. An OPTIONS request querying capabilities can hence be routed to and answered by one entity and a subsequent INVITE by a completely different one.

For all these reasons SIP UAs implementing trickle ICE SHOULD always perform half trickle, unless that behaviour is specifically overridden by configuration (which could be the case in controlled environments where every agent supports trickle ICE).

[TODO maybe define a way for GRUU supporting agents to do full trickle]

4. Encoding and Sending Candidate Information

Trickled candidates and end-of-candidates indications sent by trickle ICE SIP UAs are transported as payload in SIP INFO requests sent within the already established dialog. Such payloads are encoded in an SDP format as specified in [I-D.ivov-mmusic-trickle-ice].

Since neither the "a=candidate" nor the "a=end-of-candidates" lines contain information matching them to a stream, this is handled through the use of MID [RFC3388] as follows:

```
INFO sip:alice@example.com SIP/2.0
...
Info-Package: trickle-ice
Content-type: application/sdp
Content-Disposition: Info-Package
Content-length: ...

a=mid:1
a=candidate:1 1 UDP 1658497328 192.168.100.33 5000 typ host
a=candidate:2 1 UDP 1658497328 96.1.2.3 5000 typ srflx
a=m-line-id:2
a=candidate:2 1 UDP 1658497328 96.1.2.3 5002 typ srflx
a=end-of-candidates
```

5. Info Package

5.1. Overall Description

This specification defines an INFO package meant for use by SIP user agents implementing Trickle ICE. Typically INFO requests would carry ICE candidates discovered after the user agent has sent or received a trickle-ice offer.

5.2. Applicability

The purpose of the ICE protocol is to establish a media path. The candidates that this specification transports in INFO requests are part of this establishment. There is hence no way for them to be transported through the not yet existing media path.

Candidates sent by a trickle ICE agent after the offer, are meant to follow the same signalling path and reach the same entity as the offer itself. While it is true that GRUUs can be used to achieve this, one of the goals of this specification is to allow operation of trickle ICE in as many environments as possible including those with no GRUU support. Using out-of-dialog SUBSCRIBE/NOTIFY requests would not satisfy this goal.

5.3. INFO Package Name

This document defines a SIP INFO Package as per [RFC6086]. The INFO Package token name for this package is "trickle-ice"

5.4. INFO Package Parameters

This document does not define any INFO package parameters.

5.5. SIP Option-Tags

[RFC6086] allows Info Package specifications to define SIP option-tags. This document therefore stipulates that SIP entities that support trickle ICE and this specification MUST place the 'trickle-ice' option-tag in a SIP Supported header field.

When responding to, or generating a SIP OPTIONS request a SIP entity MUST also include the 'trickle-ice' option-tag in a SIP Supported header field.

5.6. INFO Message Body Parts

Entities implementing this specification MUST include SDP encoded ICE candidates in all SIP INFO requests. The MIME type for the payload

MUST be of type 'application/sdp' as defined in Section 4 and [I-D.ivoov-mmusic-trickle-ice].

6. Example Flows

A typical successful (half) trickle ICE exchange with SIP would look this way:

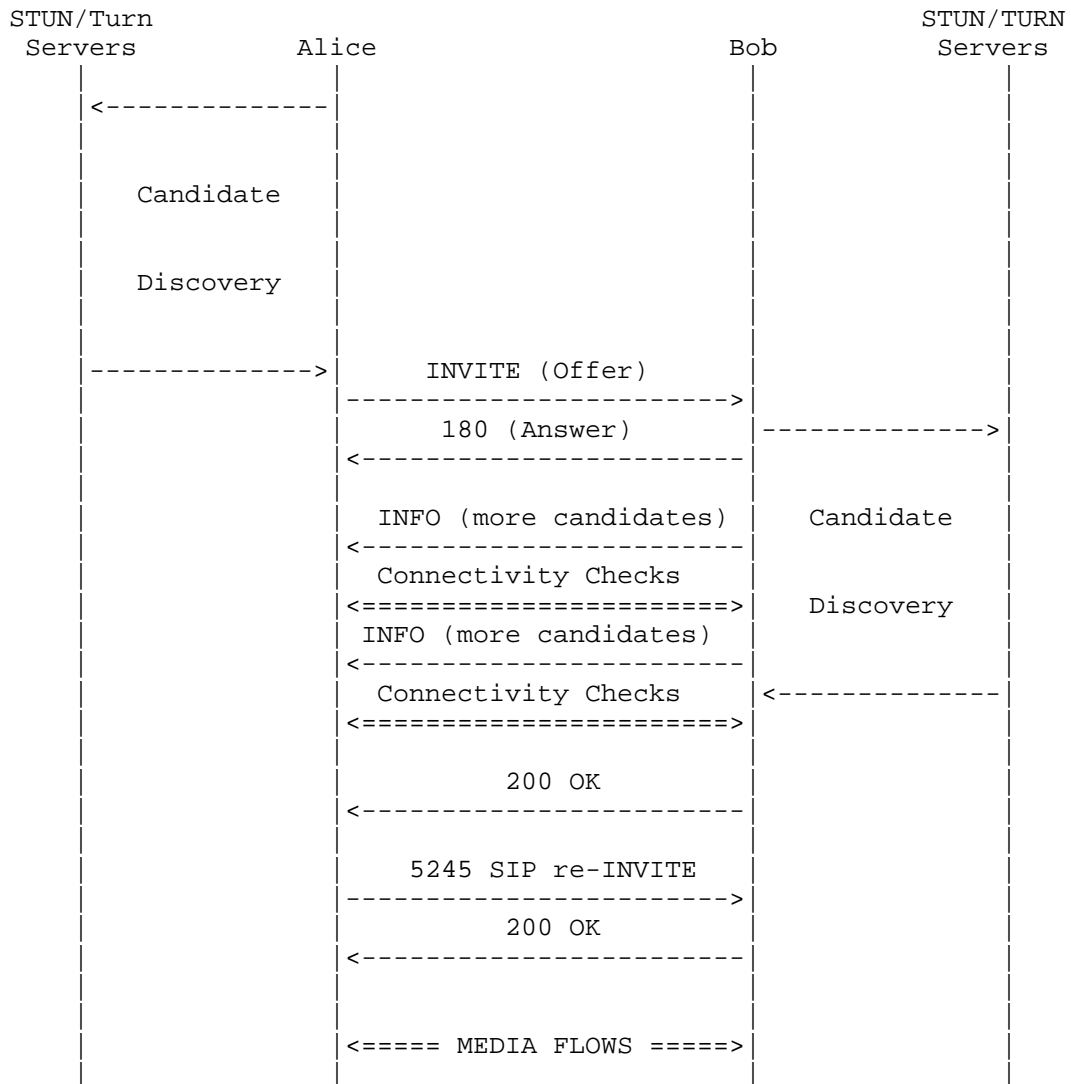


Figure 1: Example

7. Security Considerations

[TODO]

8. Acknowledgements

[TODO]

9. References

9.1. Normative References

- [I-D.ivov-mmusic-trickle-ice]
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE:
Incremental Provisioning of Candidates for the Interactive
Connectivity Establishment (ICE) Protocol",
draft-ivov-mmusic-trickle-ice-00 (work in progress),
January 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.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment
(ICE): A Protocol for Network Address Translator (NAT)
Traversal for Offer/Answer Protocols", RFC 5245,
April 2010.
- [RFC6086] Holmberg, C., Burger, E., and H. Kaplan, "Session
Initiation Protocol (SIP) INFO Method and Package
Framework", RFC 6086, January 2011.

9.2. Informative References

- [I-D.keranen-mmusic-ice-address-selection]
Keraenen, A. and J. Arkko, "Update on Candidate Address
Selection for Interactive Connectivity Establishment

(ICE)", draft-keranen-mmusic-ice-address-selection-01 (work in progress), July 2012.

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [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.
- [RFC3388] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.
- [RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

Appendix A. Open issues

At the time of writing of this document the authors have no clear view on how and if the following list of issues should be address here:

1. Should we allow for full trickle if support can be verified automatically and confirmed for a gruue with [RFC3840].
2. Can we pick between MID and stream indices for stream identification.

Authors' Addresses

Emil Ivov
Jitsi
Strasbourg 67000
France

Phone: +33 6 72 81 15 55
Email: emcho@jitsi.org

Enrico Marocco
Telecom Italia
Via G. Reiss Romoli, 274
Turin 10148
Italy

Email: enrico.marocco@telecomitalia.it

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

MMUSIC
Internet-Draft
Intended status: Informational
Expires: January 10, 2014

J. Marcon
R. Ejzak
Alcatel-Lucent
July 09, 2013

MSRP over WebRTC data channels
draft-marcon-msrp-over-webrtc-data-channels-00

Abstract

The Real-Time Communication in WEB-browsers (RTCWeb) working group is charged to provide protocols to support direct interactive rich communication using audio, video, and data between two peers' web-browsers. For the support of data communication, the RTCWeb working group has in particular defined the concept of bi-directional data channels over SCTP, where each data channel might be used to transport other protocols, called sub-protocols. This document specifies how the Message Session Relay Protocol (MSRP) can be instantiated as a WebRTC data channel sub-protocol, using the SDP offer/exchange to negotiate out-of-band the sub-protocol specific parameters. Two network configurations are documented: a WebRTC end-to-end configuration (connecting two MSRP over data channel endpoints), and a gateway configuration (connecting an MSRP over data channel endpoint with an MSRP over TCP endpoint).

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 January 10, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	3
3. Terminology	3
4. WebRTC Data Channels	4
5. End-to-end configuration	5
5.1. Support for SDP-based sub-protocol negotiation	5
5.1.1. SDP syntax	5
5.1.1.1. Channel-specific setup parameters	5
5.1.1.2. Sub-protocol specific attributes	6
5.1.2. Procedures	7
5.1.2.1. Opening a data channel	7
5.1.2.2. Closing a data channel	7
5.2. Support for MSRP data channels	7
5.2.1. Overview	7
5.2.2. MSRP URI	8
5.2.3. Session negotiation	8
5.2.4. Session opening	8
5.2.5. Data sending and reporting	8
5.2.6. Session closing	9
5.3. Support for MSRP File Transfer function	9
6. Gateway configuration	9
7. Security Considerations	10
8. IANA Considerations	10
9. Acknowledgments	10
10. References	10
10.1. Normative References	10
10.2. Informative References	11
Authors' Addresses	12

1. Introduction

The Message Session Relay Protocol (MSRP) [RFC4975] is currently defined to work over TCP connections.

The RTCWeb working group has defined the concept of bi-directional data channels running on top of SCTP/DTLS. Each data channel consists of paired SCTP streams sharing the same SCTP Stream Identifier. Data channels are created by endpoint applications through the WebRTC API, and can be used to transport proprietary or well-defined protocols, which in the latter case can be signaled by the data channel "sub-protocol" parameter, conceptually similar to the WebSocket "sub-protocol". However, apart from the "sub-protocol" value transmitted to the peer, RTCWeb leaves open how endpoint applications can agree on how to instantiate a given sub-protocol on a data channel, whether it be in-band or out-of-band (or both). As an example, the SDP offer generated by the browser includes no channel-specific information.

MSRP is a protocol for transmitting a series of related instant messages in the context of a session. In addition to instant messaging, MSRP can also be used for image sharing or file transfer.

Defining the MSRP as a data channel sub-protocol has many benefits:

- o provide to WebRTC applications a proven protocol enabling instant messaging, file transfer, image sharing
- o integrate those features with other RTCWeb voice and video features
- o leverage the SDP-based negotiation already defined for MSRP
- o allows the interworking with MSRP endpoints running on a TCP connection

This document defines the use MSRP of over WebRTC data channels, where one MSRP endpoint is an MSRP WebRTC application and the other endpoint is either an MSRP WebRTC application or an MSRP TCP application.

2. Conventions

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

3. Terminology

This document uses the following terms:

Data channel: A bidirectional channel consisting of paired SCTP outbound and inbound streams.

In-band: transmission through the peer-to-peer SCTP association.

Out-of-band: transmission through the WebRTC signaling path, using JSEP [I-D.ietf-rtcweb-jsep] and the SDP Offer/Answer model [RFC3264].

MSRP data channel: A data channel specifically used to transport the messages of one MSRP session.

Peer: From the perspective of one of the agents in a session, its peer is the other agent. Specifically, from the perspective of the SDP offerer, the peer is the SDP answerer. From the perspective of the SDP answerer, the peer is the SDP offerer.

4. WebRTC Data Channels

This section summarizes how WebRTC data channels work in general.

A WebRTC application creates a data channel via the WebRTC Data Channel API, by providing a number of setup parameters (sub-protocol, label, reliability, order of delivery, priority).

The browser then opens in-band the data channel using the DATA CHANNEL OPEN message defined in [draft-jesup-rtcweb-data-protocol]. This message carries some of the channel-specific parameters passed by the application (sub-protocol, label, reliability, order of delivery).

In case an SCTP association is already established, the browser transmits immediately the DATA CHANNEL OPEN message to the peer, on an unused SCTP stream.

In case no SCTP association is established, the browser triggers for an SDP offer/answer exchange, and sends the DATA CHANNEL OPEN message(s) once the SCTP association is established (i.e. subsequently to the reception of the answer).

The SDP offer generated by the browser is as per [draft-ietf-mmusic-sctp-sdp]. In brief, it contains one m-line for the SCTP association on top of which data channels will run, and one attribute per protocol assigned to the SCTP ports:

```
m=application 54111 DTLS/SCTP 5000 5001 5002
c=IN IP4 79.97.215.79
a=sctpmap:5000 webrtc-datachannel 16
```



```
a=sctpmap:5001 bfcf 2
a=sctpmap:5002 t38 1
```

Note: A WebRTC browser will only create an sctpmap attribute for the webrtc-datachannel protocol, and will not create sctpmap attributes for other protocols such as bfcf or t38. This example shows the hypothetical power of the syntax to support multiplexing of SCTP associations for different protocols on the same DTLS connection.

Note: this SDP syntax does not contain any channel-specific information.

5. End-to-end configuration

This section describes the network configuration where each MSRP endpoint is a WebRTC endpoint, running MSRP over an SCTP/DTLS connection.

5.1. Support for SDP-based sub-protocol negotiation

In the default procedures described in Section 4, the channel-specific parameters are notified in-band to the peer, rather than negotiated with the peer. Also, no mechanism is defined to transmit subprotocol-specific parameters to the peer.

This section defines a means to negotiate channel-specific and subprotocol-specific parameters, using the out-of-band SDP offer/exchange.

5.1.1. SDP syntax

The SDP only contains the declaration of data channels for which an SDP-based negotiation is required, and that are either being created or already opened.

5.1.1.1. Channel-specific setup parameters

For each of these data channels, the SDP lists one attribute line providing the Stream Identifier, sub-protocol, label, reliability, order of delivery, priority.

```
a=webrtc-DataChannel:5000 stream=2;label="channel 2"; \
  subprotocol="file transfer";max_retr=3
```

NOTE: the related SDP syntax has to be imported from version 3 of [draft-ietf-mmusic-sctp-sdp].

This line MUST be replicated without changes in the SDP answer, if the answerer accepts the offered data channel.

This line MUST be replicated without changes in any subsequent offer or answer, as long as the data channel is still opened at the time of offer or answer generation.

The Sub-protocol, label, reliability and order of delivery parameters MUST be equal to those transmitted in-band in the DATA CHANNEL OPEN message. The Stream Identifier MUST be equal to the SCTP Stream Identifier on which the DATA CHANNEL OPEN message is sent.

5.1.1.2. Sub-protocol specific attributes

In the SDP, each data channel declaration MAY also be followed by other SDP attributes specific to the sub-protocol in use. Each of these attributes is represented by one new attribute line, and it includes the contents of a media-level SDP attribute already defined for use with this (sub)protocol in another IETF specification.

Each sub-protocol specific attribute such as "a=accept-types:text/plain" that would normally be used to negotiate an instance of MSRP is replaced with an attribute of the form "a=wdcsa:sctp-port:stream-id original-attribute", where wdcsa stands for "webrtc-DataChannel sub-protocol attribute", sctp-port is the sctp port number assigned for webrtc-DataChannel on the media line, stream-id is the sctp stream id assigned to this instance of MSRP, and original-attribute represents the contents of the MSRP related attribute to be included.

```
a=webrtc-DataChannel:5000 stream=2;label="channel 2"; \
  subprotocol="MSRP";max_retr=3
a=wdcsa:5000:2 accept-types:text/plain
```

Thus the attribute "a=wdcsa:5000:2 accept-types:text/plain" specifies that this instance of MSRP on stream id 2 accepts plain text files.

As opposed to the data channel setup parameters, these parameters are subject to offer/answer negotiation.

The same syntax applies to any other SDP attribute required for negotiation of this instance of the sub-protocol.

5.1.2. Procedures

5.1.2.1. Opening a data channel

Opening a data channel is done in-band by the DATA CHANNEL OPEN message. However when the sub-protocol requires an SDP-based negotiation, applications **MUST NOT** send data on this channel till both SDP negotiation and DATA CHANNEL OPEN message sending are done, which may happen in any order.

When the application creates a new data channel (requiring some sub-protocol specific negotiation), the browser follows in any case a generic behavior:

- o if no SCTP association is established, the browser triggers the SDP negotiation, and sends the DATA CHANNEL OPEN message once the answer is received and the SCTP association initialized.
- o if an SCTP association is established, the browser does not trigger any SDP negotiation but instead immediately sends a DATA CHANNEL OPEN message. The application then initiates a new offer/answer exchange

Note: in this case, as the DATA CHANNEL OPEN message is sent before the offer is created, Stream ID conflicts between offers sent to the peer, and DATA CHANNEL OPEN messages received from the peer should not occur.

The application has the task to complete the browser-generated offer (or answer) with the data channel and subprotocol specific parameters in scope of the SCTP m-line. The browser is expected to ignore those parameters when the completed offer (or answer) is applied locally.

5.1.2.2. Closing a data channel

Closing a data channel is done in-band by the SSN reset mechanism, and does not trigger a new offer/answer exchange.

5.2. Support for MSRP data channels

5.2.1. Overview

This document defines how MSRP can be used as a WebRTC sub-protocol, where the MSRP-related negotiation is done as part of the SDP-based data channel negotiation defined in Section 5.1.1.2.

In this design, the MSRP connection maps to the SCTP association and the port assigned to data channels, and each MSRP session maps to one data channel exactly.

5.2.2. MSRP URI

This document extends the MSRP URI syntax [RFC3986] by defining the new transport parameter value "dc":

```
transport = "tcp" / "dc" / 1*ALPHANUM
```

5.2.3. Session negotiation

Using the syntax `a=webrtc-DataChannel:<port> <param=value>`, the SDP declaration of a given MSRP data channel can include at least all the following well-known parameters:

- o defined in [RFC4975]: "path", "accept-types", "accept-wrapped-types", "max-size"
- o defined in [RFC4566]: "sendonly", "recvonly", "inactive", and "sendrecv"
- o defined in [RFC6135]: "setup"
- o defined in [RFC6714]: "msrp-cema"
- o defined in [RFC5547]: all the parameters related to MSRP file transfer

5.2.4. Session opening

The MSRP session is normally opened by the active MSRP endpoint, which sends an MSRP SEND message (empty or not) to the other MSRP endpoint. The active MSRP endpoint does not use the path attribute to open a transport connection to its peer. Instead, the active MSRP endpoint uses the DataChannel established for this MSRP session by the procedures in Section 5.1. The cema attribute is implicitly associated with every MSRP session using data channel transport.

5.2.5. Data sending and reporting

5.2.6. Session closing

Either endpoint can close the MSRP session by closing the underlying data channel. Closing an MSRP session should not trigger an SDP negotiation.

5.3. Support for MSRP File Transfer function

[RFC5547] defines an end-to-end file transfer method based on MSRP and the SDP offer/answer mechanism. This file transfer method is also usable by MSRP WebRTC endpoints, with the following considerations:

- o As an MSRP session maps to one data channel, a file transfer session maps also to one data channel.
- o SDP attributes specified in [RFC5547] for a file transfer m-line are embedded as subprotocol-specific attributes as defined in Section 5.1.1.2.
- o Each file chunk is transmitted over one SCTP user message.
- o Once the file transfer is complete, the same data channel MAY be reused for another file transfer.
- o Following the aborting of a file transfer, the SDP can be updated by adding the "inactive" attribute to the list of subprotocol-specific attributes associated with the corresponding data channel.

6. Gateway configuration

This section describes the network configuration where one endpoint runs MSRP over a WebRTC SCTP/DTLS connection, the other MSRP endpoint runs MSRP over one or more TLS/TCP connections, and the two endpoints interwork via an MSRP gateway.

Specifically, a gateway can be configured to interwork an MSRP session using a data channel with a peer that does not support data channel transport in one of two ways. In one model, the gateway performs as a MSRP B2BUA to interwork all the procedures as necessary between the endpoints. No further specification is needed for this model.

Alternately, the gateway can use CEFA procedures to provide transport level interworking between MSRP endpoints using different transport protocols as follows.

When the gateway performs transport level interworking between MSRP endpoints, all of the procedures in section Section 5.1 apply to each peer, with the following additions:

- o The endpoint establishing an MSRP session using data channel transport shall not request inclusion of any relays, although it may interoperate with a peer that signals the use of relays.
- o The gateway receiving an SDP offer that includes a request to negotiate an MSRP session on a data channel can provide transport level interworking in the same manner as a CEMA SBC by forwarding TCP or TLS transport parameters in a new m line with the appropriate attributes within the forwarded SDP offer.
- o Similarly, a gateway receiving an SDP offer to negotiate an MSRP session using TCP or TLS transport with an endpoint that only supports data channel transport for MSRP can provide transport level interworking in the same manner as a CEMA SBC by establishing a new data channel for the MSRP session with the target endpoint.

7. Security Considerations

To be completed.

8. IANA Considerations

To be completed.

9. Acknowledgments

The authors wish to thank... for their invaluable comments.

10. References

10.1. Normative References

- [I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-02 (work in progress), October 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

- [RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.
- [RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for the Message Sessions Relay Protocol (MSRP)", RFC 4976, September 2007.
- [RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.
- [RFC6135] Holmberg, C. and S. Blau, "An Alternative Connection Model for the Message Session Relay Protocol (MSRP)", RFC 6135, February 2011.
- [RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.

10.2. Informative References

- [I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "RTCWeb Data Channels", draft-ietf-rtcweb-data-channel-04 (work in progress), February 2013.
- [I-D.jesup-rtcweb-data-protocol] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Protocol", draft-jesup-rtcweb-data-protocol-04 (work in progress), February 2013.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [WebRtcAPI] Bergkvist, A., Burnett, D., Narayanan, A., and C. Jennings, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120821, August 2012, <<http://www.w3.org/TR/2012/WD-webrtc-20120821>>.

Authors' Addresses

Jerome Marcon
Alcatel-Lucent
Route de Villejust
Nozay 91620
France

Email: jerome.marcon@alcatel-lucent.com

Richard Ejzak
Alcatel-Lucent
1960 Lucent Lane
Naperville, Illinois 60563-1594
US

Phone: +1 630 979 7036

Email: richard.ejzak@alcatel-lucent.com

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: January 03, 2014

R. Penno, Ed.
P. Martinsen
D. Wing
A. Zamfir
Cisco
July 02, 2013

Meta-data Attribute signalling with ICE
draft-martinsen-mmusic-malice-00

Abstract

It can be useful for applications to provide flow metadata information to on-path devices to influence flow treatment in the network. Provided that the network is able to provide useful feedback, this can also influence path selection if an application have multiple flow paths to choose from.

This draft describes how this can be achieved by adding metadata to the STUN packets sent during the ICE connectivity checks or a slightly modified version of the keep-alive mechanism. Devices on the media path can use the metadata information to prioritize the flow, perform traffic engineering, or provide network analytics and notifications as requested by the endpoints. On-path devices can append or modify the existing metadata information in the STUN/ICE messages to enable feedback to other on-path devices or the applications in both ends of the media session.

This document describes a framework mechanism for how such metadata can be transported by STUN when ICE is in use and it covers the endpoint and on path device processing. The functionality described here is referred to as MALICE.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 January 03, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Problem Statement	3
2. Terminology	4
3. Overview of MALICE	5
3.1. Metadata Attributes	6
3.1.1. Sending and Receiving	6
3.1.2. Directionality and Asymmetry	7
3.1.3. Network Element Processing	8
3.1.4. MALICE Client and Server Processing	8
3.2. Connectivity Checks	8
3.2.1. MALICE to non-MALICE	9
3.2.2. MALICE to MALICE	9
3.3. Keepalives	10
3.4. Aggressive Nomination	10
3.5. Implications on Concluding ICE	11
3.6. Lite Implementations and MALICE	12
4. Performing Connectivity Checks	12
4.1. MALICE Client Procedures	12
4.1.1. Building the MALICE Request	12
4.1.2. Processing MALICE Responses	13

4.2.	MALICE Network Element Procedures	14
4.2.1.	Adding a new Metadata IE	14
4.2.2.	Removing a Metadata IE	16
4.2.3.	Changing a metadata IE	18
4.2.4.	Network Element Response Change	19
4.2.5.	Solving Conflicts in Metadata Attribute Values	19
4.2.6.	Conflict Resolution	22
4.3.	MALICE Server Procedures	23
5.	Concluding MALICE Processing	23
6.	Subsequent Connectivity Checks	24
7.	Security Considerations	24
7.1.	STUN Inspection	24
7.2.	Authentication	25
8.	STUN Extensions	25
8.1.	New Attributes	25
9.	IANA Considerations	26
9.1.	STUN Attribute TLV Definitions	26
9.1.1.	MD-AGENT Attribute	26
9.1.2.	MD-RESP-UP and MD-RESP-DN Attributes	26
9.1.3.	MD-PEER-CHECK Attribute	27
9.2.	Metadata Attributes sub-TLV Definitions	27
9.2.1.	FLOWDATA Request	27
9.2.2.	FLOWDATA Response	29
9.2.3.	Usage Example	31
10.	Acknowledgements	31
11.	References	32
11.1.	Normative References	32
11.2.	Informational References	32
	Authors' Addresses	32

1. Problem Statement

In the context of Content, Mobile, Fixed Service, Service Providers, Enterprise and Private networks have a need to prioritize packet flows end-to-end. These flows are often dynamic, time-bound, encrypted, peer-to-peer, possibly asymmetric, and might have different priorities depending on network conditions, direction, time of the day, dynamic user preferences and other factors. These factors may be time variant, and thus need to be signalled. Moreover, in many cases of peer-to-peer communication, flow information is known only to the endpoint. These considerations, coupled with the trend to use encryption for browser-to-browser communication [I-D.ietf-rtcweb-security-arch], imply that access lists, deep packet inspection and other static prioritization methods cannot be employed successfully to prioritize packet flows. It can also be useful for the endpoints to provide flow metadata and receive network feedback in order select an optimal media communication path. This specification describes how these problems can be solved at

different points in the network by using either STUN [RFC5389] packets sent during ICE's [RFC5245] connectivity check phase during establishment of a media session, or as part a slightly modified keep-alive mechanism after the session is established. Devices on the media path can use the metadata information to prioritize the flow, perform traffic engineering, or provide network analytics and notifications as requested by the endpoints. On-path devices can append or modify the existing metadata information in the STUN/ICE messages. The ICE agents may use this information to learn about the status of their requests at on-path devices.

This document describes a framework mechanism for how such metadata can be transported by STUN when ICE is in use with UDP based media and it covers the endpoint and middlebox processing. The functionality described here is referred to as MALICE.

2. Terminology

Metadata - Information and actions associated with a flow but not used for matching. For example, firewall and NAT actions, application name, Diffserv marking actions, media-type, amongst others.

Flow - 5-tuple composed on source and destination IP addresses, IP protocol, source and destination ports.

MALICE Agent - An ICE agent [RFC5245] that supports this specification

MALICE Check - An ICE connectivity check that includes client metadata and that may include the results from network elements that have processed the request.

MALICE Message - An ICE connectivity check message (STUN Binding request or response) that carries metadata attributes.

Metadata Attribute - A STUN attribute that contains a set of information elements in the form of type-length-values (TLVs).

Information Elements - Information elements (IE) are TLVs that contain the actual metadata such as minimum bandwidth, delay tolerance, firewall action, etc.

Network Elements - Devices such as middleboxes, routers, Wireless Access LAN controller, amongst others. The terms network element and node are used interchangeably in the text.

3. Overview of MALICE

In a typical ICE deployment there are two endpoints, known as agents in ICE terminology, that attempt ICE message exchanges in order to discover one or more paths over which they can send and receive media. The ICE exchange protocol is defined in [RFC5245]. This specification proposes an extension to the ICE protocol that allows applications to request services from the network, and learn about the status of these requests and of the media paths they use. This is achieved by signaling flow and network metadata attributes between endpoints and network elements (NEs).

The means by which an implementation determines the metadata IEs to be signaled is out of the scope of this specification. Section 9 covers different scenarios where metadata may be of use. This specification defines three types of transaction that can be signaled by a MALICE agent and acted upon by NEs.

- o Binding Transaction (REQ-RESP): Endpoint requests flow prioritization, e.g. by signaling the desired service class (Section 9) that includes the minimum and maximum bandwidth, loss and delay tolerance. The following are examples of services that could be offered by network elements:
 - * IntServ: Network elements on path may perform admission control against the desired service class. If resources are not available, a middlebox may return an error (or allocated BW = 0) or it may try to admit the flow in a lower service class. In the latter case, the middlebox will update the response with the new service class. If resources are available, they are allocated for the flow and guaranteed (in a stable network) for the lifetime of the flow.
 - * DiffServ: A middlebox may perform flow classification. Flows are guaranteed QoS as long as there is no oversubscription. If the corresponding service queue becomes full, drops and delays affect all flows in that service class.
- o Advisory Transaction (REQ-RESP):
 - * Notification Subscription: An endpoint may request the network to send notifications when certain conditions occur. One example described in Section 9 is notification when congestion is about to occur in the class of service associated with the flow. Other services in this category may be defined in the future.

- * Query : Endpoints may request information from the network. One example described in Section 9 is an endpoint requesting the currently available bandwidth, delay and loss tolerance of the service class associated with the flow. Network elements update the response STUN attributes if local values are more restrictive than the ones carried in the message. At the end of the request/response check, the endpoint has the information about the end-to-end b/w, delay and loss characteristics of the path.
- o Informational Transaction (INFO-ONLY):
 - * Endpoints send INFO-ONLY attributes to describe their flows. This service can be used in managed environments like enterprise or data center.

The following new comprehensive-optional STUN attributes are defined in order to support this functionality:

- o MD-AGENT: includes client agent metadata information for the flow described by the 5-tuple identified in the STUN/ICE header.
- o MD-RES-UP: contains the result of the request processing by the network elements on upstream path.
- o MD-RES-DN: includes the result of the request processing by the network elements on downstream path.
- o MD-PEER-CHECK-RES: contains the result of the MALICE check performed by the peer agent.
- o MD-INFO: contains flow descriptive information.

The client agent includes a combination of MD-AGENT, MD-RESP-UP and MD-RESP-DN to create one of the three transaction types described above. In addition, the FLOWDATA sub-TLV is defined to support flow prioritization through a Binding Transaction.

3.1. Metadata Attributes

The main focus of this specification is around the services described in the previous section which are implemented through REQ-RESP attribute signaling. For these services, most of the actions described here apply.

3.1.1. Sending and Receiving

Sending metadata can be done early in the connectivity check phase of ICE [RFC5245] section-7 and the result of metadata processing may be taken into account by the controlling agent during the nomination process. Once a candidate pair is selected to be used for media, MALICE agents use the consent freshness mechanism described in [I-D.muthu-behave-consent-freshness] to signal metadata attributes.

If a server agent supports MALICE, it MUST reflect back in the STUN Binding Response message the metadata attributes that were received in the STUN Binding Request. It is up to the server agent whether to use the metadata present in the binding request for its own purposes, for example adjusting the metadata it will put in its own binding request.

Network Elements on the path that are MALICE capable may intercept and read the metadata attributes from the connectivity or consent freshness checks. They may also update the message with the result of a REQ-RESP request. When doing so, the NEs MUST NOT add significant delay while attribute processing is in progress and SHOULD wait for the next refresh message for result update.

3.1.2. Directionality and Asymmetry

It is important to mention that some attributes may be bidirectional in nature, while others may be associated with a given direction. A bi-directional attribute is represented by individual upstream and downstream attributes.

In order to take into account directionality and routing asymmetry the following rules are proposed for the STUN Binding request/response messages used in connectivity check and consent freshness mechanism:

STUN Request On-path devices only process upstream attributes and if necessary update the original request message with the result.

STUN Response On-path devices only process downstream attributes and if necessary update the original response message with the result.

Due to asymmetric routing, a NE may see only binding request or response messages for a given candidate pair and therefore it may read and process metadata for upstream only, downstream only or both. In some cases, upstream and downstream paths may span the same node but over different interfaces and in this case a middlebox may need to use different ingress and/or egress interface policies for the two directions of the media.

3.1.3. Network Element Processing

When processing MALICE messages, NEs generally perform the following steps:

1. Intercept and read the metadata attributes from the connectivity or consent freshness checks.
2. Depending on the metadata information elements carried in the message and on the current state (e.g. resource availability, policies, etc.), a node may perform certain actions (e.g. install local policies for the flow described by the message, start monitoring the flow, perform marking, etc.).
3. If the results of these actions are readily available, the network element should include them in the currently intercepted message. Otherwise any required response is conveyed in the next refresh message.
4. Forwards the MALICE message downstream.

The current specification makes sure that network elements do not have to change the STUN message size, instead the MD-RESP-* attributes are inserted as place holders for updates from network.

3.1.4. MALICE Client and Server Processing

The MALICE client agent includes metadata information elements in the new MD-AGENT STUN attribute defined in this specification. The MD-AGENT attribute MUST be included before INTEGRITY. If a response is required for all or a subset of these information elements, the client agent may also include the new MD-RESP-DN (before INTEGRITY) and MD-RESP-UP (after INTEGRITY) as place holders that can be used by on-path devices to provide a response.

When a MALICE server agent receives a Binding Request, it copies the MD-AGENT and the MD-RESP-UP TLV in the response, adds the INTEGRITY attribute and then inserts the MD-RESP-DN attribute to be filled by on path nodes for the downstream direction. When forming the response (success or error), the agent running the server follows the rules of Section 6 of [RFC5389]. It MUST NOT send an 'Error Response' message class if the processing of metadata attributes is the only one that has failed. Instead the MALICE error indications are included in the MD-RESP-UP to communicate to the client the success/error indications for the metadata processing.

3.2. Connectivity Checks

Connectivity checks are extended by this specification to include metadata attributes in both request and response messages. In the presence of REQ-RESP metadata attributes, a MALICE agent may consider the connectivity check successful if responses for the check received indicate success. It is not necessary that the metadata attribute results, if present, also indicate success.

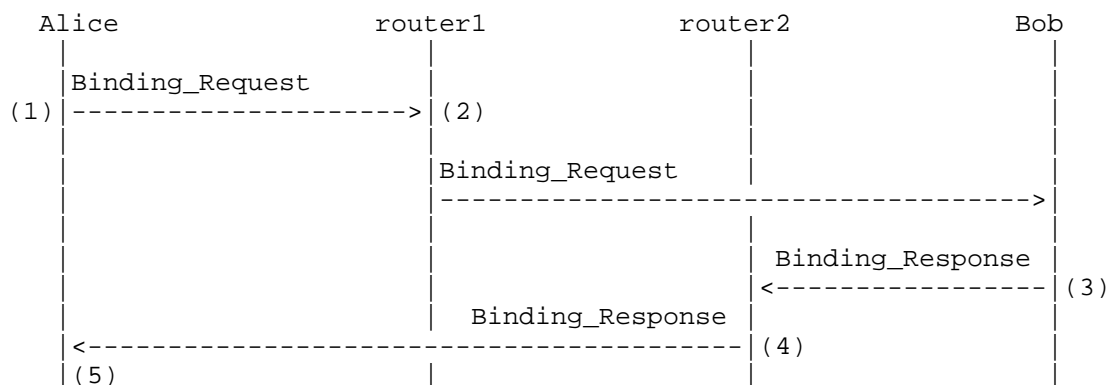
The MALICE Server agent MAY also include the new MD-PEER-CHECK-RES TLV defined in this specification if it has already performed a MALICE check and has the result available. This is useful if the MALICE Server is the controlled agent and wishes to influence the nomination process at MALICE Client (controlling agent).

3.2.1. MALICE to non-MALICE

A MALICE client agent does not have prior knowledge if the peer supports this specification. If the peer agent is not MALICE capable, it will not reflect back the metadata STUN attributes. Therefore a MALICE client agent will know if peer is MALICE capable after the first exchange of the connectivity check. The client may choose to continue to signal the metadata attributes to benefit from possible upstream network element processing but should not expect any results from the network.

3.2.2. MALICE to MALICE

A remote MALICE agent echoes back in the Binding Response message all metadata received in the request. In the example below MALICE upstream network elements (router1 in the diagram below) processes MD-AGENT and MD-RESP-UP attributes present in the STUN binding request while MD-AGENT and MD-RESP-DOWN attributes present in the STUN binding response are processed by network elements (router2) in the downstream path.



FLOW-METADATA MALICE to MALICE

1. Alice creates a Binding Request, adds MD-AGENT and result (MD-RESP-UP and MD-RESP-DN) attributes with desired metadata information elements.
2. Router1 inspects the Request message and, if allowed (based on realm, security and policy considerations), reads MD-AGENT attribute and its information elements. If the result of processing is available, router1 writes the result in the MD-RESP-UP attribute. It then forwards the request.
3. Bob processes the Binding Request as described in the ICE RFC [RFC5245](Section 7.2). When Bob builds the response, it copies the metadata attribute MD-AGENT and the MD-RESP-UP attributes into the Binding Response and adds MD-RESP-DN after the integrity attribute. Bob then transmits the message.
4. Router2 (first MALICE network element for the downstream direction) inspects the Response message, reads the metadata attribute and MAY change the result (MD-RESP-DN) including the local results if available. It then transmits the message.
5. When Alice receives the Binding Response message, the same processing described in ICE RFC [RFC5245] (Section 7.1.3) applies. Then it extracts the metadata upstream and downstream attributes. If Alice's agent has the controlling role, it may take into account this information during the candidate pair selection step (if this check was part of the initial connectivity check sequence).

3.3. Keepalives

This specification proposes the use of consent freshness messages [I-D.muthu-behave-consent-freshness] in place of indications in order to have up to date results on the MALICE checks used by media. This is required since network conditions may change during the lifetime of a flow resulting in changes, including new failure indications, in MALICE responses.

3.4. Aggressive Nomination

With aggressive nomination, the controlling agent includes the nominated flag in every connectivity check it sends for all media components. Once the first check for a component succeeds, it is added to the valid list with the nominated flag set. The nominated candidate pair may start being used by the media at any time after. This lowers the chance of MALICE results to be collected. Therefore,

if the controlling MALICE agent expects to consider the metadata attribute processing result into the candidate pair selection process, it SHOULD NOT use aggressive nomination. The controlled MALICE agent does not have a way to influence the peer with respect to the nomination procedure used. If the peer is non-MALICE, the agent SHOULD NOT signal any MD attributes. If a MALICE agent chooses to use the aggressive nomination, the endpoints should be prepared for transient candidate selection as described in Section 8.1.1.2 of [RFC5245]. Using aggressive nomination is an implementation trade-off between quick call initiation versus waiting to determine the best path (using regular nomination and waiting until MALICE checks finish).

3.5. Implications on Concluding ICE

When the MALICE client agent receives the STUN binding response it extracts the metadata results. A controlling agent may choose to ignore the received metadata information or consider it in the decision process. The figure below shows MALICE used in a regular nomination process.

```

L(Malice)                                R(Malice)
-----                                -----

    <----- STUN request + {MDrl(i)}      \  R's
STUN response ----->                    /  check
+ {MDrl(i)}

                                local result: MDrl

STUN request + {MDlr(i)} ----->        \  L's
    <----- STUN response                /  check
    + {MDlr(i)}
    + MDrl (result)

local result: MDlr
e2e result: comp(MDlr, MDrl)

STUN request + {MDlr(i)} + flag ---->    \  L's
    <----- STUN response                /  check
    + {MDlr(i)}
```

Notations:

L is the controlling agent.

{MDrl(i)} is the set of metadata attributes sent from R to L in the request. In the (2nd, 3rd,...) response back they will also include the result. Similar notation for the checks in the other direction.

MDrl is a an overall success/fail type of indication for the MALICE check R->L

comp(MDlr, MDrl) - is a function that determines the overall end to end MALICE result based on both local check result and the one from the peer.

If a connectivity check response is received for an already nominated pair, the controlling agent may inform the application but MUST NOT restart the nomination process. In the case where the result of a MALICE check is not available in the response at the time of nomination, any subsequent MALICE results become informative.

3.6. Lite Implementations and MALICE

As described in [RFC5245], lite ICE implementations do not send connectivity checks but only reply to them. A lite ICE implementation may be extended to become a lite MALICE implementation by adding the functionality associated with the MALICE Server. When a lite MALICE server agent receives a STUN binding request, it copies the metadata related attributes as described in earlier sections. A lite MALICE implementation will never include an MD-PEER-CHECK-RES attribute in the STUN binding response, since it never runs ICE or MALICE checks.

4. Performing Connectivity Checks

This section describes how MALICE agents perform connectivity checks and how network elements process and modify the information in the connectivity check messages.

4.1. MALICE Client Procedures

4.1.1. Building the MALICE Request

This section describes how STUN and ICE are extended to include metadata attributes and refers to them in generic terms. The new attributes and their usage defined in Section 9 are included in the connectivity checks performed by MALICE agents.

The Client agent starts the connectivity check by sending a STUN binding request following the procedures described in Section 7.1.2 of [RFC5245]. A MALICE client MAY include metadata attributes in the request. The way the application determines the attributes to be

sent to the MALICE agent for signaling is outside the scope of this specification. The client agent may reduce the attribute set based on other factors (e.g. MTU considerations).

The client encodes metadata information in the MD-AGENT attribute. It then builds the MD-RESP-UP and MD-RESP-DN attributes, including an information element for each REQ-RESP attribute for which a response is desired. The values in these IEs are initialized as described in the corresponding metadata information element section. MD-AGENT and MD-RESP-DN MUST be included before INTEGRITY, and MD-RESP-UP after INTEGRITY so that it can be changed by on-path devices.

4.1.2. Processing MALICE Responses

A MALICE agent processes a STUN binding response and depending on the presence of metadata attributes, their contents, and the procedures of [RFC5245] section 7.1.3.1 the result of MALICE connectivity check is considered unknown, failure or success as described below

4.1.2.1. Unknown

If the STUN response message does not include any metadata related STUN attributes, this is an indication that the peer is not MALICE capable. In this case the client should change the pair state to Succeeded.

It is possible that the STUN Client receives a response that includes metadata STUN attributes, but doesn't include any valid results from NEs or STUN Server. This can happen if NEs are not MALICE enabled.

4.1.2.2. Failure

In the presence of a MALICE peer, a MALICE check is considered failed if either of the following is true:

- o the ICE check has failed as described in Section 7.1.3.1 of [RFC5245].
- o the client determines that the metadata included by an on-path device in the Binding response does not meet its criteria for success. The success criteria is application dependent and outside the scope of this specification.

4.1.2.3. Success

A MALICE check is considered successful if all of the following are true:

- o the ICE check as described in Section 7.1.3.1 of [RFC5245] has succeeded.
- o the Binding response indicates that MALICE NEs have satisfactorily processed all the RESP-REQ information elements.

4.2. MALICE Network Element Procedures

A MALICE network element intercepts ICE request and response messages, reads metadata information from the MD-AGENT attribute and triggers corresponding processing. When the result of this processing is available, the MALICE node MAY update the MD-RESP-xx attribute carried in the message. As a consequence, it is recommended (and stated [RFC5245]) that the agent perform a few identical checks in order to allow NEs to react to and communicate the result of the metadata processing.

MALICE NEs consume router resources to maintain per flow state and, depending on the information elements and requests, to enforce per flow QoS or perform monitoring. State and associated attributes are considered alive as long as periodic refresh messages that include those attributes are received. In the absence of refreshes [I-D.muthu-behave-consent-freshness] or if attributes cease to be present in those refreshes, attributes time out, associated resources are released and state may be removed.

MALICE agents can signal the same metadata information elements for a flow. Therefore it is possible that different STUN messages types containing the same information elements, with same or different values, are seen by NEs. It is also possible that the two agents signal different metadata for the same flow.

During the lifetime of a session, agents can change the values of information elements, remove or add new IEs. It is also possible that a NE changes the result values over the lifetime of a session. A NE should determine if a newly intercepted STUN message indicates a refresh versus a change as compared to the previously intercepted message. A refresh resets the lifetime of an IE and state. A change indicates if new IEs are being created or if existing ones are being modified or removed.

4.2.1. Adding a new Metadata IE

When a new IE is signaled in a STUN message, a network element should create state for the flow if not already present, and trigger any required processing. If the network element, while processing the metadata attribute, will add significant delay and cause timeouts in the agent state machines, it is recommended that it forwards the STUN message and use the next refresh message to provide the results. When the next STUN message is received, the NE should provide the result of processing this information element only if the locally stored (and acted upon) value is the same as the one in the newly received message. Otherwise a removal or modification has occurred.

The diagram below illustrates the exchange and processing when a new IE is added. Alice sends a STUN request upstream with attribute MD-RESP-UP, MD-AGENT and IE X=A. The network element creates the f(L,R) state where it stores the requested metadata value (m: X=A), the context it was received from (s: MALICE request) and the result of processing (r: x=N). It then updates the response attribute MD-RESP-UP in the STUN request with X=N and forwards it to Bob. Bob reflects back the original metadata requested value and the result.

Alice(L)	NE	Bob(R)
-----	---	-----
Alice's STUN Request		
x=A for Upstream (L->R)		
IE x=A		IE x=A
resp x=<>		resp x=N
-----> ----->		
f(L,R): create:		
m: x=A, s: req		
r: x=N		
<-----.....<-----		
		IE x=A
		resp x=N

Upstream Attribute Initial Signaling

Similar processing happens for downstream attributes except that the NE's actions (intercept, flow state creation, etc.) happen when a STUN response is intercepted.

There are many possible transaction types for "X=A". For example:

- o Endpoint requests a particular service: "Reserve BW=5Mbps", the endpoint requests a 5Mbps reservation.
- o Endpoint requests network notification: "Notify if BW < 5Mbps", the endpoint requires a notification when the queue capacity used for this flow falls below the 5Mbps limit.
- o Endpoint request statistics for the flow path: "BW=<>", where <> is the unspecified value for attribute BW, the endpoint requires a response with the current available queue capacity used for this flow.

It is assumed in the rest of this specification that the attribute, information element and/or context unambiguously identify the actions required at network element.

4.2.2. Removing a Metadata IE

Flow state and all its metadata ages out and should be removed when the state has not been refreshed recently by a request or response message. The way to determine the timeout interval is described in [I-D.muthu-behave-consent-freshness].

In addition, metadata must be immediately deleted and associated resources released if the IE is not present in any subsequent messages for the flow. An IE should be considered stale and removed if it ceases to appear in STUN requests or responses (section 3.1.2) having the same 5-tuple flow. As illustrated in the diagram below, a NE implementation should keep track of the source and value of the IEs received and detect per source addition, change and removal. More details are provided in the next sections. In the diagram below Bob's messages do not go through the NE element:

1. Alice signals metadata X=A for the first time. Actions are described in the previous section.
2. Bob signals the same value and equivalent direction for X and in his STUN request, this is copied in the STUN Response from Alice to Bob. When the NE intercepts this L->R response message, it extracts X=A, retrieves the existing information f(L,R) and adds MALICE Response as a new source.
3. Alice sends a new check without any metadata attributes. The NE retrieves the f(L,R) state and removes the MALICE Request from the source list. The flow state is maintained as the NE still sees refreshes for X in the L->R responses to Bob's checks.

4. Bob sends a new STUN connectivity check without any attributes. The NE retrieves the f(L,R) state and removes the MALICE Response from the source list. Since X has no source, it also removes X from the metadata information element list and releases any resources associated with X. And because the flow state has no more attributes, it also removes the state.

```

Alice(L)           NE           Bob(R)
-----           ---           -----

Alice's STUN Request (1)
  x=A for Upstream (L->R)

IE   x=A           IE   x=A
resp x=<>           resp x=N
----->           ----->
                        f(L,R): create:
                          m: x=A, s: req
                          r: x=N

<-----.....<-----
                        IE   x=A
                        resp x=N

                        Bob's STUN Request (2)
                        x=A for Downstream (L->R)

                        IE   x=A
                        resp x=<>
<-----.....<-----

IE   x=A           IE   x=A
resp x=<>           resp x=N
----->           ----->
                        f(L,R): update
                          a: x=A, s: req
                          x=A, s: resp
                          r: x=N

Alice's STUN Request (3)
  no attributes
----->           ----->
                        f(L,R): update
                          a: x=A, s: resp
                          r: x=N

```

```

<-----.....-----

                                Bob's STUN Request  (4)
                                no attributes
<-----.....<-----
----->----->
f(L,R): update
a: <none>, s:<none>
r: x=N
f(L,R): release resources for X
remove state

```

Upstream Attribute Removal

4.2.3. Changing a metadata IE

It is possible for a client to change an IE value. Every request/response message contains an MD-RESP-xx attribute with "not specified" values when sent from the agent. In other words, the agent does not include the result from previous check. When a node detects a change in an attribute value it should trigger the appropriate actions. Like in the case of initial attribute creation, the node should provide the answer in the next refresh message if the answer is not immediately available.

In the diagram below, Alice changes the value of information element X from A to B in the second STUN request which causes the network element to provide a different response.

```

Alice(L)          NE          Bob(R)
-----          ---          -----

Alice's STUN Request                                (1)
x=A for Upstream (L->R)

IE   x=A          IE   x=A
resp x=<>          resp x=N
----->----->
f(L,R): create
m: x=A, s: req
r: x=N

<-----.....-----
                                IE   x=A
                                resp x=N

```

Alice's STUN Request (2)
 x=B for Upstream (L->R)

```

IE    x=B                      IE    x=B
resp x=<>                      resp x=M
----->                      ----->
                                f(L,R): update
                                m: x=B, s: req
                                r: x=M
<-----.....
                                IE    x=B
                                resp x=M

```

Upstream Attribute Change

4.2.4. Network Element Response Change

It is possible that the network element result of processing of an IE changes as resource availability changes, e.g. new links are added and removed, new flows come and go, etc. For example, a NE can change the bandwidth available for a flow and may need to update the MD-RESP-xx attribute if the local value is more restrictive (e.g. less bandwidth, lower delay tolerance, etc.) than the one included in the message. Again, it is important for this node to check that the MD-AGENT attribute includes the same attribute and value for which the answer is provided.

4.2.5. Solving Conflicts in Metadata Attribute Values

A conflict in a metadata information element occurs when the two agents signal different values for same IE and for the same direction of the flow.

A conflict occurs for an IE X in the upstream direction if the values of X in the L check request are different than in the R check response. When a NE detects an IE conflict it SHOULD keep both values. If the IE is part of binding request, the MALICE node must perform conflict resolution as described in the diagram below and act on the result.

1. Alice sends a request for X with value A for the upstream direction. The NE intercepts the message, creates f(L,R) state and stores X=A remembering this was received in Alice's request. The NE then determines that the response to A should be N, therefore it updates the STUN message and forwards it to Bob.

2. Bob sends a request for X with value B for the upstream direction. The NE intercepts the response for the Bob->Alice request, extracts X=B from the response, looks up f(L,R) flow state, stores (x=B, s:resp) and determines that a conflict has occurred for attribute X since (x=A, s: req) is present in the state. The NE runs the conflict resolution and determines that x=B should be the value used, determines that the result of processing B is M, updates the STUN response and forwards the response to Bob.
3. When the next refresh for X with value A is received from Alice, the NE updates the result to M and forwards the request to Bob. Bob reflects back the result in the response and Alice receives the changed result.

```

Alice(L)           NE           Bob(R)
-----           --           -----

Alice's STUN Request (1)
x=A for Upstream (L->R)

IE  x=A           IE  x=A
resp x=<>         resp x=N
----->         ----->
                        f(L,R): create:
                        m: x=A, s: req
                        r: x=N

<-----.....<-----
                        IE UP(x=A)
                        resp UP(x=N)

                        Bob's STUN Request (2)
                        x=A for Downstream (L->R)

                        IE  x=B
                        resp x=<>
<-----.....<-----

IE  x=B           IE  x=B
resp x=<>         resp x=M
----->         ----->
                        f(L,R): update
                        m: x=A, s: req
                        x=B, s: resp
                        <- conflict detected!

```

```

        <- resolution x=B
r: x=M

```

```

Alice's STUN Request                                     (3)
  x=A for Upstream (L->R)

```

```

IE   x=A                                         IE   x=A
resp x=<>                                         resp x=M
----->                                         ----->
      f(L,R): refresh:
        m: x=A, s: req
          x=B, s: resp
        r: x=M

<-----.....-----
                                attr UP(x=A)
                                resp UP(x=M)

```

Upstream Attribute Conflict

Note that for INFO-ONLY and ADVISORY transactions a conflict resolution cannot occur and, therefore, results should be kept per source. Typical NE resources allocated for these attributes are monitors created to detect conditions or collect network statistics. It is up to the implementation to decide on what can be shared in terms of resources in this case. In the diagram below, for illustration purposes, a second monitor is created for Bob's notification request.

```

Alice(L)          Mid          Bob(R)
-----          ---          -----

Alice's STUN Request                                     (1)
  Notif for UP BW < 10Mbps

IE   bw=10Mbps                                         IE   bw=10M
resp bw=<>                                         resp bw=<>
----->                                         ----->
      f(L,R): create:
        m: bw=10M, s: req
        r: bw=<>, start monitor

<-----.....-----
                                attr bw=10M
                                resp bw=<>

```

Alice's STUN Request (2)
First refresh after condition

```

IE    bw=10Mbps                IE bw=10Mbps
resp bw=<>                    resp bw=8Mbps
----->                    ----->
f(L,R): create:
  m: bw=10Mbps, s: req
  r: bw=8Mbps, keep monitor

<-----.....<-----
                                IE bw=10Mbps
                                resp bw=8Mbps

```

Bob's STUN Request (3)
x=A for Downstream (L->R)

```

                                IE    bw=6Mbps
                                resp bw=<>
<-----.....<-----

IE    bw=6Mbps                IE bw=6Mbps
resp bw=<>                    resp bw=<>
----->                    ----->
f(L,R): update
  m: bw=10Mbps, s: req
  r: bw=8Mbps, keep monitor
  m: bw=6Mbps, s: resp
  r: bw=<>, start monitor2

```

Network Analytics and Notifications

4.2.6. Conflict Resolution

The definition/description of an information element must include a description of how conflict resolution should be done by network elements. Below are a few examples:

- o Informational only transactions: the IEs included are signaled in the upstream direction only and they are processed by middleboxes on path with the STUN request. They should never generate conflicts.
- o Binding transactions (QoS): the following attributes are currently defined:

- * Bandwidth: UP/DOWN Max Bandwidth, UP/DOWN Min Bandwidth
- * Service Class: UP/DOWN Delay, Loss and Jitter tolerance - specified as: 0=undefined, 1=very low, 2=low, 3=medium, 4=high
- * Priority: UP/DOWN DSCP

For all these attributes the conflicts are resolved by choosing the less strict values (apply a MIN function). For example, assume Alice and Bob request the same service class. If Alice requests 10Mbps UP bandwidth, Bob requests 5Mbps DOWN bandwidth and there are 7Mbps available for the service class specified in the request, the middlebox should allocate 5Mbps and update the result in Alice's check STUN Response. If Alice and Bob request different service classes, the less restrictive is first selected and then the MIN function is applied to the bandwidth values.

- o Advisory transactions (Network Analytics): there should not be any conflict resolution applied to these attributes. It is perfectly valid for Alice to request different network analytics than Bob or different thresholds for congestion notifications. As shown in the previous diagram, middleboxes should keep track of the different sources for a given attribute and, in case of network attributes, keep per source results and maybe resources.

4.3. MALICE Server Procedures

When the Malice Server agent receives a STUN Request it follows the same rules described in Section 7.2 of [RFC5245]. In addition, when building the STUN Response the following rules MUST be followed:

- o MD-AGENT and MD-RESP-UP attributes are inserted before INTEGRITY
- o If the result of the local MALICE check is present, an MD-PEER-CHECK-RES attribute with the result is included before INTEGRITY
- o A copy of the MD-RESP-DN attribute received in the STUN Request is included unmodified after INTEGRITY

5. Concluding MALICE Processing

A MALICE Controlling agent is expected to run regular nomination only. This specification also reinforces the recommendation to run a number of checks before nominating a pair. This increases the probability of receiving network element and peer MALICE responses and therefore having more information for the nomination process.

When nominating a pair, the controlling agent may consider the MALICE information received in the last STUN Response and give preference to the pair whose connectivity check indicated favorable network conditions.

6. Subsequent Connectivity Checks

It is possible for a MALICE Client to request a service and include metadata attributes after the nomination process. It is also possible that a successful MALICE check for the nominated (active) pair fails during the media session lifetime. The MALICE Client will have at all times the current status of the MALICE check for the active pair. The actions that the client takes when these change are currently out of the scope of this document. In the absence of support for other specification, these MALICE check status changes are informative only.

7. Security Considerations

7.1. STUN Inspection

Network elements processing STUN packets are open to denial of service attacks from endpoints when there is no previous authorization and indication of which STUN messages should be inspected. The vulnerability and attack vector is similar to those documented for the IP router alert option in [RFC6398].

Flooding a NE with bogus (or simply undesired) STUN messages that contain metadata could impact its operation in undesirable ways. For example, if the NE punts the datagrams containing STUN messages to the slow path, such an attack could consume a significant share of the NE's slow path and could also lead to packet drops in the slow path (affecting operation of all other applications and protocols operating in the slow path), thereby resulting in a denial of service (DoS) [RFC4732]. Like with other protocols, it is recommended that network elements that implement this functionality use rate limited queues when punting STUN messages. In addition, it is recommended that the implementation enforces limits on the number of states created by the MALICE connectivity checks.

However, the main issue is that the STUN message does not provide a convenient universal mechanism to accurately and reliably distinguish between interesting and unwanted messages. This, in turn, creates a security concern when the STUN metadata attribute is used, because, short of appropriate network element- implementation-specific mechanisms, the NE slow path is at risk of being flooded by unwanted traffic.

One solution to this problem is to include a precursor authorization step where a third-party device authorizes the endpoint and populates the NE with 5-tuple information of the packet carrying the STUN message. [TODO: Reference third party authorization draft]

7.2. Authentication

While endpoints are able to authenticate STUN messages received by a peer endpoint, network elements are unable to authenticate STUN messages. Further, endpoints are not fully trusted by network elements, so network elements need some assurance that what is signaled has been authorized by an application server that defines policies or attributes for a given media flow. Even if an endpoint is well-behaved, the network elements need a means of ensuring STUN messages are not altered during transmission.

8. STUN Extensions

8.1. New Attributes

This specification defines five new attributes, MD-AGENT, MD-REALM, MD-RESP-UP, MD-RESP-DN and MD-PEER-CHECK.

- o The MD-AGENT is inserted in the Binding request by the client agent and copied in the Binding response by the server agent. It includes the flow metadata generated by the client agent.
- o The MD-RESP-UP is inserted by the client agent in the Binding request and updated by MALICE nodes on upstream path. A MALICE server agent copies this attribute in the response message.
- o The MD-PEER-CHECK attribute is inserted by the MALICE server agent in the response message and includes the result of the MALICE check executed by the server agent.
- o The MD-RESP-DN is inserted by the client agent in the Binding request, copied by the MALICE server agent in the response and updated by MALICE nodes on downstream path.

In addition, two new sub-TLVs are defined to provide flow prioritization service. This specification allows for easy addition of IEs in the future.

- o FLOWDATA Request sub-TLV is included in the MD-AGENT STUN attribute and indicates the desired flow treatment

- o FLOWDATA Response sub-TLV is included in the MD-RESP-* STUN attributes and indicates, when received by the client in the STUN Binding Response, the result of the processing

9. IANA Considerations

This specification registers five new STUN attributes. All attributes include metadata informational elements. Section 10.2 describes a possible STUN specific encoding for these. Another proposal can be found in [I-D.draft-flow-metadata-encoding] and [I-D.draft-flow-metadata-framework]

9.1. STUN Attribute TLV Definitions

This section registers four new STUN attributes per the procedures in [RFC5389].

```
0x0C02: MD-AGENT
0x0C03: MD-RESP-UP
0x0C04: MD-RESP-DN
0x0C05: MD-PEER-CHECK
```

9.1.1. MD-AGENT Attribute

Metadata attributes are encoded in sub-TLV format with each sub-TLV corresponding to an information element or metadata. Section 10.3 describes in detail the information elements that can be included in the MD-AGENT attribute. When parsing the STUN request and response, the MD-AGENT STUN attribute Length should be used to identify the location of next STUN attribute.

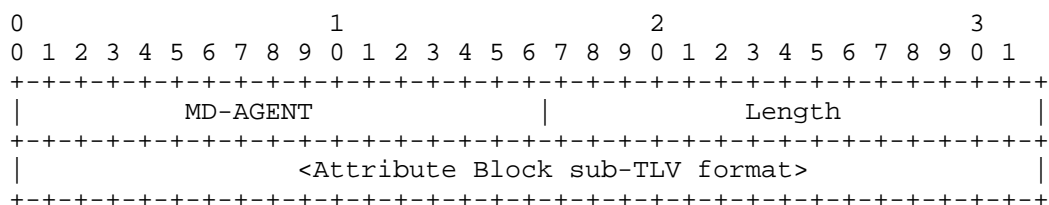


Figure 1: MD-AGENT Attribute

9.1.2. MD-RESP-UP and MD-RESP-DN Attributes

Network Metadata attributes are encoded in sub-TLV format with each sub-TLV corresponding to an information element or metadata.

Section 10.3 describes in detail the network information elements that can be included. When parsing the STUN request and response, the MD-RESP-XX STUN attribute Length should be used to identify the location of next STUN attribute.

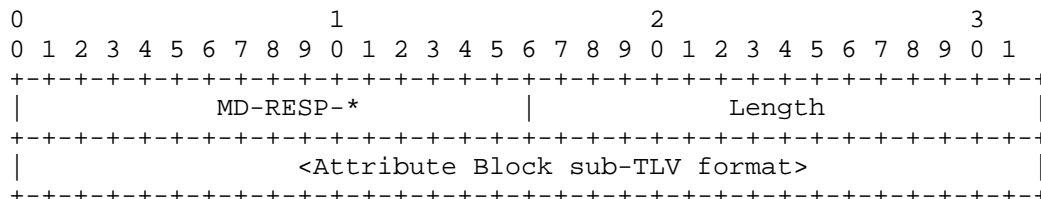


Figure 2: MD-RESP- Attribute

Where MD-RESP-* = {MD-RESP-UP | MD-RESP-DN}

9.1.3. MD-PEER-CHECK Attribute

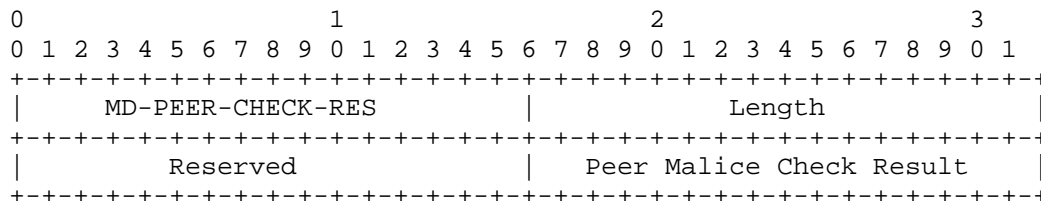


Figure 3: MD-PEER-CHECK Attribute

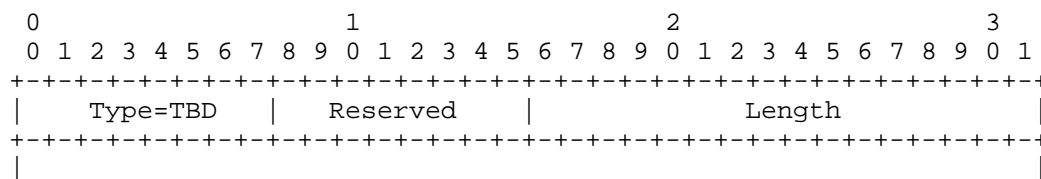
Peer Malice Check Result - "Success" or "Failure".

9.2. Metadata Attributes sub-TLV Definitions

Metadata information elements are encoded in sub-TLV format and included in MD-AGENT and MD-RESP-* STUN attributes described earlier.

9.2.1. FLOWDATA Request

The FLOWDATA IE has the following format.



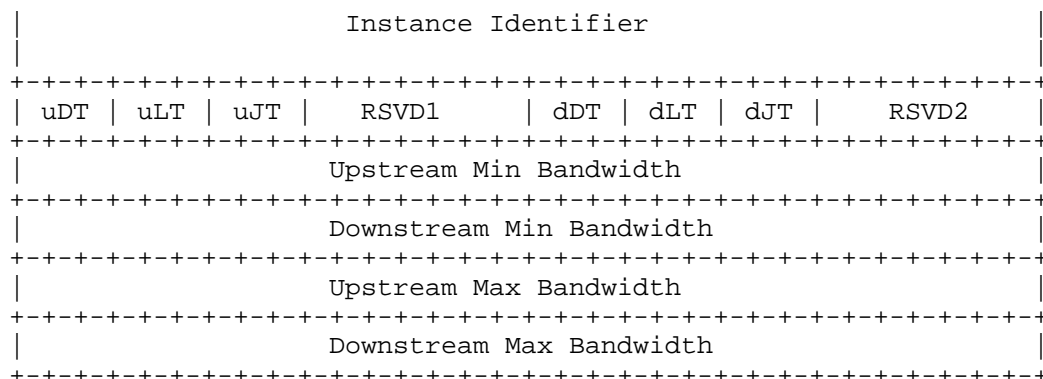


Figure 4: FLOWDATA Request

Type: TBD (optional to process)

Reserved: Must be 0 and ignored by the server.

Length: Option Length is 32 octets.

May appear in: STUN/ICE Binding Request and Response, inside the MD-AGENT STUN attribute

Maximum occurrences: 1

Description of the fields:

Instance Identifier: Instance identifier, see below for description.

uDT: Upstream Delay Tolerance, 0 means no information is available.
1=very low, 2=low, 3=medium, 4=high.

uLT: Upstream Loss Tolerance, 0 means no information is available.
1=very low, 2=low, 3=medium, 4=high.

uJT: Upstream Jitter Tolerance, 0 means no information is available.
1=very low, 2=low, 3=medium, 4=high.

RSVD1: Reserved (7 bits), MUST be ignored on reception and MUST be 0 on transmission

dDT: Downstream Delay Tolerance, 0 means no information is available. 1=very low, 2=low, 3=medium, 4=high.

dLT: Downstream Loss Tolerance, 0 means no information is available. 1=very low, 2=low, 3=medium, 4=high.

dJT: Downstream Jitter Tolerance, 0 means no information available.
1=very low, 2=low, 3=medium, 4=high.

RSVD2: Reserved (7 bits), MUST be ignored on reception and MUST be 0 on transmission.

Upstream Minimum Bandwidth Minimum Upstream bandwidth in bytes per second, 0 means no information is available.

Downstream Minimum Bandwidth: Minimum Downstream bandwidth in bytes per second, 0 means no information is available.

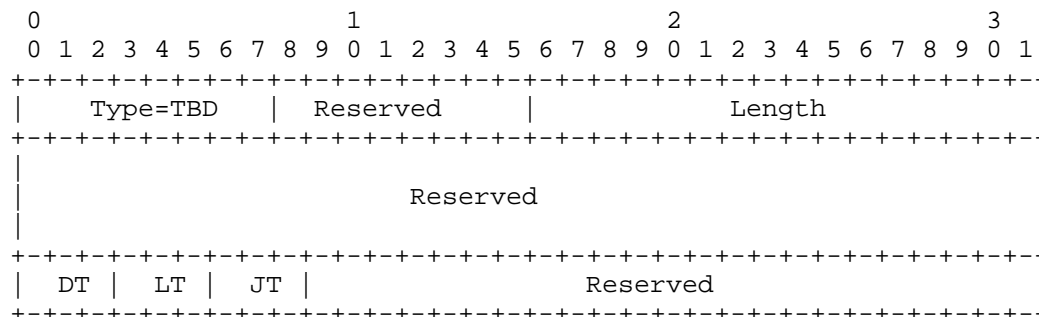
Upstream Maximum Bandwidth: Maximum Upstream bandwidth in bytes per second, 0 means no information is available.

Downstream Maximum Bandwidth: Maximum Downstream bandwidth in bytes per second, 0 means no information is available.

The instance identifier accommodates network traffic where multiple 5-tuples exist for a particular data flow, but the bandwidth flows only over the aggregate of the multiple 5-tuples. One example of this are a phone call which rings on two phones. Only one of those phones will answer first (and send data). FLOWDATA is signaled for both of those phone's IP addresses and ports, using the same Instance Identifier, indicating to the network that the flow data is being shared with those two different 5-tuples. Another example is TCP video streaming which retrieves short pieces of the movie, often over separate TCP connections for load balancing, which would use the same Instance Identifier for each TCP connection. The way the instance identifier is determined is out of the scope of this document.

9.2.2. FLOWDATA Response

This IE is meant for responses from network to endpoint. It can be included in MD-RESP-UP or MD-RESP-DN, therefore indicating the direction for which the response applies.



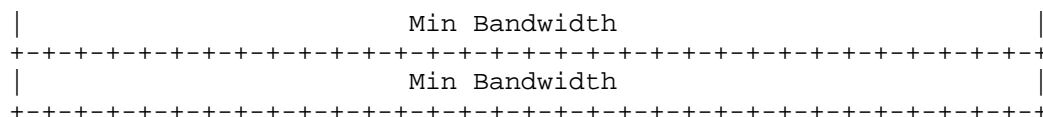


Figure 5: FLOWDATA Response

Type: TBD (optional to process)

Reserved: Must be 0 and ignored by the server.

Length: Option Length is 24 octets.

May appear in: STUN/ICE Binding Request and Response, inside the MD-RESP-UP and/or MD-RESP-DN STUN attributes.

Maximum occurrences: 1

When included in MD-RESP-UP TLV the FLOWDATA Response indicate the response from middleboxes that are on the upstream path. When included in MD-RESP-DN TLV the FLOWDATA Response indicate the response from middleboxes that are on the downstream path.

Description of the fields:

Reserved: 96 bits, MUST be ignored on reception and MUST be 0 on transmission.

DT: Delay Tolerance, 0 means no information is available.

LT: Loss Tolerance, 0 means no information is available.

JT: Jitter Tolerance, 0 means no information is available.

Reserved: Reserved (7 bits), MUST be ignored on reception and MUST be 0 on transmission

Minimum Bandwidth Minimum bandwidth in bytes per second, 0 means no information is available.

Maximum Bandwidth: Maximum bandwidth in bytes per second, 0 means no information is available.

9.2.3. Usage Example

This section describes how the STUN protocol elements defined above are used to implement flow prioritization.

- o Endpoint Metadata Request (REQ-RESP) - Flow Prioritization:
Endpoint asks flow prioritization by including in the Binding request non-0 values in the FLOWDATA Request and values initialized to 0 in MD-RESP-UP and MD-RESP-DN TLVs. Upstream MALICE nodes update the MD-RESP-UP with the results. Peer includes in the Binding response the received MD STUN TLVs and the MD-PEER-CHECK-RESP. Downstream MALICE nodes update the MD-RESP-DN TLV. In the example below, the endpoint received the required prioritization for the upstream direction and a lower than requested one for downstream.

* Binding Request sent by MALICE Client:

- + MD-AGENT (InstID=0, uDT=1, uLT=1, uJT=1, dDT=2, dLT=2, dJT=2, uMinBW=4mbps, uMaxBW=5mbps, uMinBW=5mbps, MaxBW=10mbps)
- + MD-RESP-UP (DT=0, LT=0, JT=0, MinBW=0mbps, MaxBW=0mbps)
- + MD-RESP-DN (DT=0, LT=0, JT=0, MinBW=0mbps, MaxBW=0mbps)

* Binding Response received by MALICE Client:

- + MD-AGENT (InstID=0, uDT=1, uLT=1, uJT=1, dDT=2, dLT=2, dJT=2, uMinBW=4mbps, uMaxBW=5mbps, uMinBW=5mbps, MaxBW=10mbps)
- + MD-ATTR-UP (DT=1, LT=1, JT=1, MinBW=4mbps, MaxBW=5mbps)
- + MD-ATTR-DN (DT=2, LT=2, JT=2, MinBW=4mbps, MaxBW=5mbps)
- + MD-PEER-CHECK-RES ("Success")

10. Acknowledgements

Authors would like to thank Paul Jones, Sergio Mena de la Cruz and Tirumaleswar Reddy for their comments and review.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4732] Handley, M., Rescorla, E., IAB, "Internet Denial-of-Service Considerations", RFC 4732, December 2006.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC6398] Le Faucheur, F., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, October 2011.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

11.2. Informational References

- [I-D.ietf-rtcweb-security-arch]
Rescorla, E., "RTCWEB Security Architecture", draft-ietf-rtcweb-security-arch-06 (work in progress), January 2013.
- [I-D.muthu-behave-consent-freshness]
Perumal, M., Wing, D., R, R., and H. Kaplan, "STUN Usage for Consent Freshness", draft-muthu-behave-consent-freshness-03 (work in progress), February 2013.

Authors' Addresses

Reinaldo Penno (editor)
Cisco Systems, Inc.
170 West Tasman Drive
San Jose 95134
USA

Email: repenno@cisco.com

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens vei 20
Lysaker, Akershus 1366
Norway

Email: palmarti@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA

Email: dwing@cisco.com

Anca Zamfir
Cisco Systems, Inc.
EPFL, Quartier de l'Innovation
Ecublens, Vaud 1015
Switzerland

Email: ancaz@cisco.com

MMUSIC Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

S. Nandakumar
Cisco
July 15, 2013

SSRC Group Based Simulcast Signaling
draft-nandakumar-mmusic-rtcweb-grouping-00

Abstract

In some applications it may be necessary to send multiple media encodings corresponding to a media source with in independent RTP media streams. This is called Simulcast. This document discusses a framework for describing simulcast media streams in SDP and also defines semantics to express relationship amongst them. The semantics defined in this document are to be used with the source specific grouping framework defined in the [RFC5576]

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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	2
2. Introduction	3
3. Solution Overview	4
4. imageattr for a=ssrc attribute	4
5. The SIMULCAST ssrc-group attribute	5
6. SDP Examples	5
7. Offer/Answer Consideration	6
8. Simulcast and Multiplexing	6
9. Simulcast under Multicast RTP Topology	6
10. IANA Considerations	7
11. Acknowledgements	7
12. Normative References	7
Author's Address	8

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Capture Device: The physical source of stream of media data of one type such as camera or microphone.

Media Source: A Media Source logically defines the source of a raw stream of media data as generated either by a single capture device or by a conceptual source. A Media Source represents an Audio Source or a Video Source.

Media Encoding: A particular encoding applied to the media data from a media source through the choice of sampling, bit-rate and other codec configuration parameters.

Media Stream: Media from a Media Source is encoded and packetized to produce one or more Media Streams representing a sequence of RTP packets.

RTP Source: Same as Media Stream.

RTP Session: An RTP session is an association among a group of participants communicating with RTP. It is a group communications channel which can potentially carry a number of Media Streams. Within an RTP session, every participant finds out meta-data and control information (over RTCP) about all the Media Streams in the

RTP session. The bandwidth of the RTCP control channel is shared within an RTP Session.

Media Transport: A Media Transport defines an end-to-end transport association for carrying one or more RTP Sessions. The combination of a network address and port uniquely identifies such a transport association, for example an IP address and a UDP port.

2. Introduction

Simulcast refers to taking media from a single media 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.

[I-D.westerlund-avtcore-rtp-simulcast] describes different encodings of a media content to be combination of the following:

- o Bit-rate: The difference is the amount of bits spent to encode the media thus giving different quality.
- o Codec: Different media codecs are used to ensure that different receivers that do not have a common set of decoders can decode at least one of the versions. This can include codec configuration options that are not compatible, like video encoder profiles, or the capability of receiving the transport packetization.
- o Sampling: Different sampling of media, in spatial as well as in temporal domain, may be used to suit different rendering capabilities or needs at the receiving endpoints, as well as a method to achieve different bit-rates. For video streams, spatial sampling affects image resolution and temporal sampling affects video frame rate. For audio, spatial sampling relates to the number of audio channels and temporal sampling affects audio bandwidth. Obviously, a difference in sampling may result in difference in bit-rate.

In any application that needs to send multiple encodings, there is a potential need for simulcast. The purpose of this document is to discuss suitable signaling solution in SDP for describing and negotiating simulcast streams, within the context of the Real Time Transport Protocol(RTP).

3. Solution Overview

The source-attribute specification [RFC5576] provides mechanisms describing Real-Time Protocol (RTP) [RFC3550] sources, identified by their synchronization source (SSRC) identifier, in the Session Description Protocol (SDP) [RFC4566], to associate attributes with these sources, and to express relationships among individual sources.

To describe and negotiate simulcast streams within SDP for a given media source, the following framework is being proposed:

- o Each physical media source is represented by its own unique m-line. This is a strict one-to-one mapping; a single media source device cannot be spread across several m-lines, nor may a single m-line represent multiple media sources.
- o Each simulcast media stream is marked with an a=ssrc attribute to correlate it with its RTP Packets
- o A new SSRC Grouping semantic is proposed to express the simulcast relationship between the media streams.
- o In the absence of the above grouping semantic, multiple SSRCs in a single m-line are interpreted as alternate sources for the same media source.
- o For multi-resolution simulcast, [RFC6236] imageattr is proposed for a=ssrc attribute line, to specify send multiple resolutions, for example.
- o For the receiver control of selecting the simulcast stream to receive, the mechanisms defined in [I-D.lennox-mmusic-sdp-source-selection] is proposed.
- o When multicast topology is used to distribute RTP/RTCP packets, having same multicast address across all the m=lines is proposed when multiplexing framework such as BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] is in operation.

Providing explicit resolutions on a per-SSRC basis for 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.

4. imageattr for a=ssrc attribute

This document extends a=ssrc attribute [RFC5576] by introducing [RFC6236] imageattr enabling specification of multi-resolution simulcast streams for a given media source.

5. The SIMULCAST ssrc-group attribute

This document also extends [RFC5576] SSRC Grouping Framework semantics (a=ssrc-group) to indicate relationship between the simulcast media streams for a given media source. A semantic called "SIMULCAST" is defined to achieve this purpose.

6. SDP Examples

The example SDP in Figure 1 shows simulcast encoding with the use of different codec-specific parameters using two different payload types for a camera source. It also describes different resolutions for each encoded simulcast media stream. The "SIMULCAST" SSRC grouping semantic is included to signal the relationship.

```
m=video 62537 RTP/SAVPF 96 97
a=rtpmap:96 VP8/90000
a=rtpmap:97 VP8/90000
a=sendrecv
a=ssrc:29154
a=imageattr:96 [x=1280,y=720]           //Stream 1
a=fmtp:96 max-fr=30;max-fs=3600
a=ssrc:47182
a=imageattr:97 [x=640,y=360]           //Stream 2
a=fmtp:97 max-fr=15;max-fs=880
a=ssrc-group:SIMULCAST 29154 47182
```

Figure 1

The SDP example in Figure 2, advertises simulcast of 2 video sources corresponding to 2 cameras, each at 2 resolutions.

```
m=video 62537 RTP/SAVPF 96 97 //Camera 1
a=rtpmap:96 VP8/90000
a=rtpmap:97 VP8/90000
a=sendrecv
a=ssrc:11111
a=ssrc:22222
a=imageattr:96 [x=1280,y=720]
a=fmtp:96 max-fr=30;max-fs=3600
a=imageattr:97 [x=640,y=360]
a=fmtp:97 max-fr=15;max-fs=880
a=ssrc-group:SIMULCAST 29154 47182
```

```
m=video 54890 RTP/SAVPF 96 97 //Camera 2
a=rtpmap:96 H264/90000
a=rtpmap:97 H264/90000
a=sendrecv
a=ssrc:33333
a=ssrc:44444
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1;max-fr=30
a=fmtp:97 profile-level-id=4d0028;packetization-mode=1;max-fr=15
a=ssrc-group:SIMULCAST 33333 44444
```

Figure 2

7. Offer/Answer Consideration

The Offer/Answer model for this specification adheres to the model as applicable to Source-Specific Media SDP attributes [RFC5576] and its extensions [I-D.lennox-mmusic-sdp-source-selection].

8. Simulcast and Multiplexing

Multiplexing in RTP can be achieved in several ways as listed:

- Payload Type based Multiplexing

- Media Stream Multiplexing based on SSRC in a Single RTP Session

- RTP Session Multiplexing over a single Media Transport.

The proposed solution in this document assumes Media Stream Multiplexing to transport the simulcast encodings from several media sources with in a single RTP Session, whenever possible. Such a solution can be envisioned by the usage of multiplexing frameworks, such as, BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation]. In the absence of such a framework, each media source gets its own RTP Session with its associated simulcast encodings multiplexed based on their respective SSRCs.

9. Simulcast under Multicast RTP Topology

[RFC5117] describes several underlying network topologies supported by RTP. In this section the operation of the solution under multicast topologies is analyzed.

The proposed solution enables right behavior for RTCP reporting across the members of the group, since all the sources and their simulcast streams form a single RTP Session under the BUNDLE multiplexing framework.

In the scenarios where using a single 5-tuple for all the media sources is not possible, carrying each source with its simulcast encodings in its own RTP Session ensures correct behavior.

Allowing media sources to send different simulcast encodings to different multicast group addresses is not directly enabled within the solution proposed. Such a situation might arise to enable the receivers to selectively choose the multicast groups based on their receiving capabilities or interests. However, the mechanisms defined in [I-D.lennox-mmusic-sdp-source-selection] enables receiver control to selectively choose the simulcast streams of interest.

10. IANA Considerations

TBD

11. Acknowledgements

I would like to thanks Cullen Jennings for his inputs and review on the proposed solution.

12. 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.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [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.
- [I-D.westerlund-avtcore-rtp-simulcast]
Westerlund, M., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP Sessions", draft-westerlund-avtcore-rtp-simulcast-02 (work in progress), February 2013.
- [I-D.lennox-raiarea-rtp-grouping-taxonomy]
Lennox, J. and K. Gross, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-lennox-raiarea-rtp-grouping-taxonomy-00 (work in progress), February 2013.
- [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-04 (work in progress), June 2013.

Author's Address

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

S. Nandakumar
Cisco
July 15, 2013

A Framework for SDP Attributes when Multiplexing
draft-nandakumar-mmusic-sdp-mux-attributes-03

Abstract

The Session Description Protocol (SDP) provides mechanisms to describe attributes of multimedia sessions and of individual media streams (e.g., Real-time Transport Protocol (RTP) sessions) within a multimedia session. In the RTCWeb WG, there is a need to use a single 5-tuple for sending and receiving media associated with multiple media descriptions ("m=" lines). Such a requirement has raised concerns over the semantic implications of the SDP attributes associated with the RTP Sessions multiplexed over a single transport layer flow.

The scope of this specification is to provide a framework for analyzing the multiplexing characteristics of SDP attributes. The specification also categorizes existing attributes based on the framework described herein.

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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	5
2. Terminology	5
3. Motivation	5
4. SDP Attribute Analysis Framework	6
5. Analysis of Existing Attributes	10
5.1. RFC4566 - SDP: Session Description Protocol	10
5.2. RFC4585 - RTP/AVPF	11
5.3. RFC5761 - Multiplexing RTP and RTCP	11
5.4. RFC4574 - SDP Label Attribute	12
5.5. RFC5432 - QoS Mechanism Selection in SDP	12
5.6. RFC4568 - SDP Security Descriptions	12
5.7. RFC5762 - RTP over DCCP	13
5.8. RFC6773 - DCCP-UDP Encapsulation	14
5.9. RFC5506 - Reduced-Size RTCP in RTP Profile	14
5.10. RFC6787 - Media Resource Control Protocol Version 2	15
5.11. RFC5245 - Interactive Connectivity Establishment (ICE)	15
5.12. RFC5285 - RTP Header Extensions	16
5.13. RFC3605 - RTCP attribute in SDP	17
5.14. RFC5576 - Source-Specific SDP Attributes	17
5.15. RFC6236 - Image Attributes in SDP	18
5.16. RFC6285 - Rapid Acquisition of Multicast RTP Sessions	18
5.17. RFC6230 - Media Control Channel Framework	19
5.18. RFC6364 - SDP Elements for FEC Framework	19
5.19. RFC4796 - Content Attribute	20
5.20. RFC3407 - SDP Simple Capability Declaration	20
5.21. RFC6284 - Port Mapping between Unicast and Multicast RTP Sessions	21
5.22. RFC6714 - MSRP-CEMA	21
5.23. RFC4583 - SDP Format for BFCP Streams	22
5.24. RFC5547 - SDP Offer/Answer for File Transfer	22
5.25. draft-ietf-mmusic-media-loopback	23
5.26. RFC5760 - RTCP with Unicast Feedback	23
5.27. RFC3611 - RTCP XR	24
5.28. RFC5939 - SDP Capability Negotiation	24
5.29. RFC6871 - SDP Media Capabilities Negotiation	25

5.30.	RFC4567 - Key Management Extensions for SDP and RTSP . . .	25
5.31.	RFC4572 - Comedia over TLS in SDP	26
5.32.	RFC4570 - SDP Source Filters	26
5.33.	RFC6128 - RTCP Port for Multicast Sessions	27
5.34.	RFC6189 - ZRTP	27
5.35.	RFC4145 - Connection-Oriented Media	28
5.36.	RFC5159 - OMA BCAST SDP Attributes	29
5.37.	RFC6193 - Media Description for IKE in SDP	29
5.38.	RFC6064 - SDP and RTSP Extensions for 3GPP	30
5.39.	RFC3108 - ATM SDP	33
5.40.	3GPP TS 24.182	34
5.41.	3GPP TS 24.183	34
5.42.	3GPP TS 24.229	35
5.43.	ITU T.38	35
5.44.	ITU-T H.248.15	36
5.45.	RFC4975 - The Message Session Relay Protocol	37
5.46.	Historical	38
6.	bwtype Attribute Analysis	38
6.1.	RFC4566 - SDP: Session Description Protocol	38
6.2.	RFC3556 - SDP Bandwidth Modifiers for RTCP Bandwidth . . .	39
6.3.	RFC3890 - Bandwidth Modifier for SDP	40
7.	rtcp-fb Attribute Analysis	40
7.1.	RFC4585 - RTP/AVPF	40
7.2.	RFC5104 - Codec Control Messages in AVPF	41
8.	rtcp-fb "ack/nack" Attribute Analysis	41
8.1.	RFC4585 - RTP/AVPF	41
8.2.	RFC6285 - Unicast-Based RAMS	42
8.3.	RFC6679 - ECN for RTP over UDP/IP	42
8.4.	RFC6642 - Third-Party Loss Report	43
9.	Codec Control Messages Analysis	44
9.1.	RFC5104 - Codec Control Messages in AVPF	44
10.	group Attribute Analysis	44
10.1.	RFC5888 - SDP Grouping Framework	44
10.2.	RFC3524 - Mapping Media Streams to Resource Reservation Flows	44
10.3.	RFC4091 - ANAT Semantics	45
10.4.	RFC5956 - FEC Grouping Semantics in SDP	45
10.5.	RFC5583 - Signaling Media Decoding Dependency in SDP . . .	46
11.	ssrc-group Attribute Analysis	46
11.1.	RFC5576 - Source-Specific SDP Attributes	46
12.	QoS Mechanism Token Analysis	46
12.1.	RFC5432 - QoS Mechanism Selection in SDP	47
13.	k= Attribute Analysis	47
13.1.	RFC4566 SDP: Session Description Protocol	47
14.	content Attribute Analysis	47
14.1.	RFC4796 - MSRP Relays	48
15.	Payload Formats	48
15.1.	RFC5109 - RTP Payload Format for Generic FEC	48

16. IANA Considerations	49
17. Security Considerations	49
18. Acknowledgments	50
19. Change Log	50
20. References	50
20.1. Normative References	50
20.2. Informative References	50
Author's Address	56

1. Introduction

Real-Time Communication Web (RTCWeb) framework requires Real-time Transport Protocol as the media transport protocol and Session Description Protocol (SDP) [RFC4566] for describing and negotiating multi-media communication sessions.

SDP defines several attributes for capturing characteristics that apply to the individual media descriptions (described by "m=" lines) and the overall multimedia session. Typically different media types (audio, video etc) described using different media descriptions represent separate RTP Sessions that are carried over individual transport layer flows. However, in the RTCWeb WG, a requirement has arisen to multiplex several RTP Sessions over a single transport layer flow. This in turn has made necessary to understand the interpretation and usage of the SDP attributes defined for the multiplexed media descriptions.

Given the number of SDP attributes registered with the IANA [IANA] and possibility of new attributes being defined in the future, there is need for generic future-proof framework to analyze these attributes for their applicability in the transport multiplexing use-cases.

The document starts with providing the motivation for requiring such a framework. This is followed by introduction to the SDP attribute analysis framework/procedures, following which several sections applies the framework to the SDP attributes registered with the IANA [IANA]

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].

3. Motivation

The time and complications of setting up ICE [RFC5245] and DTLS-SRTP [RFC5763] transports for use by RTP, and conservation of ports, forms an requirement to try and reduce the number of transport level flows needed. This has resulted in the definition of ways, such as, [I-D.ietf-mmusic-sdp-bundle-negotiation] and [I-D.ietf-avt-multiplexing-rtp] to multiplex RTP over a single transport flow in order to preserve network resources such as port numbers. This imposes further restrictions on applicability of these

SDP attributes as they are defined today.

The specific problem is that there are attribute combinations which make sense when specified on independent m-lines -- as with classical SDP -- that do not make sense when those m-lines are then multiplexed over the same transport. To give an obvious example, ICE permits each m-line to have an independently specified ice-ufrag attribute. However, if the media from multiple m-lines is multiplexed over the same ICE component, then the meaning of media-level ice-ufrag attributes becomes muddled.

As of today there are close to 250 SDP attributes registered with the IANA [IANA] and more will be added in the future. There is no clearly defined procedure to establish the validity/applicability of these attribute when used with transport multiplexing.

4. SDP Attribute Analysis Framework

Attributes in an SDP session description can be defined at the session-level and media-level. These attributes could be semantically grouped as noted below.

- o Attributes related to media content such as media type, encoding schemes, payload types.
- o Attributes specifying media transport characteristics like RTP/RTCP port numbers, network addresses, QOS.
- o Metadata description attributes capturing session timing and origin information.
- o Attributes establishing relationships between media streams such as grouping framework

With the above semantic grouping as the reference, the proposed framework classifies each attribute into one of the following categories:

NORMAL Attributes that can be independently specified when multiplexing and retain their original semantics.

In the example given below, the direction and label attributes are independently specified for audio and video m=lines. These attributes are not impacted by multiplexing these media streams over a single transport layer flow.

```
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 49172 RTP/AVP 99
a=sendonly
a=label:1
a=rtpmap:99 iLBC/8000
m=video 49172 RTP/AVP 31
a=recvonly
a=label:2
a=rtpmap:31 H261/90000
```

NOT RECOMMENDED Attributes where multiplexing is not recommended if these attributes are in use in the SDP since doing so MAY result in incorrect behaviors

Example: Multiplexing media descriptions having attribute zrtp-hash defined with the media descriptions lacking it, would either complicate the handling of multiplexed stream or fail multiplexing.

```
v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=
c=IN IP4 client.biloxi.example.com
t=0 0
m=audio 3456 RTP/AVP 97 // with zrtp
a=rtpmap:97 iLBC/8000
<allOneLine>
a=zrtp-hash:1.10 fe30efd02423cb054e50efd0248742ac7a52c8f91bc2
df881ae642c371ba46df
</allOneLine>
m=video 34567 RTP/AVP 31 //without zrtp
a=rtpmap:31 H261/90000
```

IDENTICAL Attributes that MUST be identical across all the media descriptions being multiplexed.

Attributes such as rtcp-mux fall into this category. Since RTCP reporting is done per RTP Session, there is no way to receive RTCP control data for the video m=line in the example below. Hence rtcp-mux MUST be repeated for the video m=line as well, when multiplexed.


```
v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=
c=IN IP4 client.biloxi.example.com
t=0 0
m=audio 34567 RTP/AVP 97
a=rtcp-mux
m=video 34567 RTP/AVP 31
a=rtpmap:31 H261/90000
a=rtcp-mux
```

SUM Attributes can be set as they are normally used but software using them in a multiplex case, MUST apply the sum of all the attributes being multiplexed instead of trying to use each one. This is typically used for bandwidth or other rate limiting attributes to the underlining transport.

The software parsing the SDP sample below, should use the aggregate Application Specific (AS) bandwidth value from the individual media descriptions to determine the AS value for the multiplexed session. Thus the calculated AS value would be 256+64 bytes for the given example.

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
c=IN IP4 client.biloxi.example.com
t=0 0
m=audio 49170 RTP/AVP 0
b=AS:64
m=video 51372 RTP/AVP 31
b=AS:256
```

TRANSPORT Attributes that can be set normally for multiple items in a multiplexed group but the software MUST pick just one of the attribute of the given type for use. The one chosen is the attribute associated with the "m=" lines that represents the information being used for the transport of the RTP.

In the example below, "a=crypto" attribute is defined for both the audio and the video m=lines. The one that MUST be used for the multiplexed RTP Session is the one that corresponds to m=line with mid "two" even though the audio m=line with mid "one" appears ahead of it in the SDP. This is due to BUNDLE grouping semantics [I-D.ietf-mmusic-sdp-bundle-negotiation] which mandates the values from m=line corresponding to the mid appearing first on the a=group:BUNDLE line to be considered for setting up the RTP

Transport.

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
a=group:BUNDLE two,one
m=audio 49172 RTP/AVP 99
a=mid:one
a=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:d0RmdmcmVCspeEc3QGZiNWpVLfJhQXlcfHawJSoj|2^20|1:32
a=rtpmap:99 iLBC/8000
m=video 51374 RTP/AVP 31
a=mid:two
a=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:EcGZiNWpFJhQXdspcllekcmVCNWpVLcfHawJSoj|2^20|1:32
a=rtpmap:96 H261/90000
```

SPECIAL Attributes where the text in the source draft must be consulted for further handling when multiplexed.

As an example, for the attribute extmap, the specification defining the extension MUST be referred to understand the multiplexing implications.

TBD This category defines attributes that need more information to assign an appropriate category.

The idea behind these categories is to provide recommendations for using the attributes under RTP session multiplexing scenarios.

Section 5 analyzes attributes listed in IANA [IANA] grouped under the IETF document that defines them. The "Current" column indicates whether the attribute is currently specified as:

- o S -- Session level
- o M -- Media level
- o B -- Both
- o SR -- Source-level (for a single SSRC)

5. Analysis of Existing Attributes

5.1. RFC4566 - SDP: Session Description Protocol

RFC4566 [RFC4566] defines the Session Description Protocol (SDP) that is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation

Attr Name	Notes	Current	Category
sendrecv	Not impacted	B	NORMAL
sendonly	Not impacted	B	NORMAL
recvonly	Not impacted	B	NORMAL
inactive	Not impacted	B	NORMAL
cat	Not impacted	S	NORMAL
ptime	Not Impacted	M	NORMAL
maxptime	Not Impacted	M	NORMAL
orient	Not Impacted	M	NORMAL
framerate	Not Impacted	M	NORMAL
quality	Not Impacted	M	NORMAL
rtpmap	Not Impacted	M	NORMAL
fntp	Not Impacted	M	NORMAL
keywds	Not impacted	S	NORMAL
type	Not Impacted	S	NORMAL
tool	Not Impacted	S	NORMAL
charset	Not Impacted	S	NORMAL
sdplang	Not Impacted	B	NORMAL
lang	Not Impacted	B	NORMAL

+-----+	+-----+	+-----+	+-----+	+-----+

RFC4566 Attribute Analysis

5.2. RFC4585 - RTP/AVPF

RFC4585 [RFC4585] defines an extension to the Audio-visual Profile (AVP) that enables receivers to provide, statistically, more immediate feedback to the senders and thus allows for short-term adaptation and efficient feedback-based repair mechanisms to be implemented.

Attr Name	Notes	Current	Category
rtcp-fb	Since RTCP feedback are reported per RTP Session, this attribute should be repeated across m= lines	M	IDENTICAL

RFC4585 Attribute Analysis

5.3. RFC5761 - Multiplexing RTP and RTCP

RFC5761 [RFC5761] discusses issues that arise when multiplexing RTP data packets and RTP Control Protocol (RTCP) packets on a single UDP port. It describes when such multiplexing is and is not appropriate, and it explains how the Session Description Protocol (SDP) can be used to signal multiplexed sessions.

Name	Notes	Current	Category
rtcp-mux	RTCP muxing should be repeated across all the m=lines	M	IDENTICAL

RFC5761 Attribute Analysis

5.4. RFC4574 - SDP Label Attribute

RFC4574 [RFC4574] defines a new Session Description Protocol (SDP) media-level attribute: "label". The "label" attribute carries a pointer to a media stream in the context of an arbitrary network application that uses SDP. The sender of the SDP document can attach the "label" attribute to a particular media stream or streams. The application can then use the provided pointer to refer to each particular media stream in its context.

Name	Notes	Current	Category
label	Not Impacted	M	NORMAL

RFC4574 Attribute Analysis

5.5. RFC5432 - QoS Mechanism Selection in SDP

RFC5432 [RFC5432] defines prordures to negotiate QOS mechanisms using the Session Description Protocol (SDP) offer/answer model.

Name	Notes	Current	Category
qos-mech-send	Since QOS mechanism are signaled per flow, multiplexing multiple m=lines has no impact on per m=line QOS mechanism.	B	NORMAL
qos-mech-recv	Since QOS mechanism are signaled per flow, multiplexing multiple m=lines has no impact on per m=line QOS mechanism.	B	NORMAL

RFC5432 Attribute Analysis

5.6. RFC4568 - SDP Security Descriptions

RFC4568 [RFC4568] defines a Session Description Protocol (SDP) cryptographic attribute for unicast media streams. The attribute describes a cryptographic key and other parameters that serve to

configure security for a unicast media stream in either a single message or a roundtrip exchange.

Name	Notes	Current	Category
crypto	The multiplexing scheme MUST ensure unique SSRCS across all the media lines multiplexed. In that case, cryptographic keys corresponding to the underlying transport is used.	M	TRANSPORT
crypto	If the multiplexing scheme cannot ensure unique SSRCS across all the media lines, multiplexing MUST NOT be performed.	M	NOT RECOMMENDED

RFC4568 Attribute Analysis

5.7. RFC5762 - RTP over DCCP

The Real-time Transport Protocol (RTP) is a widely used transport for real-time multimedia on IP networks. The Datagram Congestion Control Protocol (DCCP) is a transport protocol that provides desirable services for real-time applications. RFC5762 [RFC5762] specifies a mapping of RTP onto DCCP, along with associated signalling, such that real-time applications can make use of the services provided by DCCP

Name	Notes	Current	Category
dccp-service-code	If RFC 6773 is not being used in addition to RFC 5762, the port in the m= line is a DCCP port. DCCP being a connection oriented protocol, does not allow multiple connections on the same 5-tuple.	M	NOT RECOMMENDED

+-----+	+-----+	+-----+	+-----+

RFC5762 Attribute Analysis

If RFC 6773 is being used in addition to RFC 5762 and provided that DCCP-in-UDP layer has additional demultiplexing, then it may be possible to use different DCCP service codes for each DCCP flow, given each uses a different DCCP port. Although doing so might conflict with the media type of the m= line. None of this is standardised yet and it wouldn't work as explained. Hence multiplexing MUST NOT be performed even in this alternate scenario.

5.8. RFC6773 - DCCP-UDP Encapsulation

RFC6773 [RFC6773] document specifies an alternative encapsulation of the Datagram Congestion Control Protocol (DCCP), referred to as DCCP-UDP. This encapsulation allows DCCP to be carried through the current generation of Network Address Translation (NAT) middleboxes without modification of those middleboxes

Name	Notes	Current	Category
dccp-port	Multiplexing MUST NOT be performed due to potential conflict between the port used for DCCP en/decapsulation and the RTP.	M	NOT RECOMMENDED

RFC6773 Attribute Analysis

5.9. RFC5506 - Reduced-Size RTCP in RTP Profile

RFC5506 [RFC5506] discusses benefits and issues that arise when allowing Real-time Transport Protocol (RTCP) packets to be transmitted with reduced size.

Name	Notes	Current	Category
rtcp-rsize	RTCP reduced size MUST be repeated across all the m=lines	M	IDENTICAL

```

|-----|-----|-----|-----|
+-----+-----+-----+-----+

```

RFC5506 Attribute Analysis

5.10. RFC6787 - Media Resource Control Protocol Version 2

The Media Resource Control Protocol Version 2 (MRCPv2) allows client hosts to control media service resources such as speech synthesizers, recognizers, verifiers, and identifiers residing in servers on the network. MRCPv2 is not a "stand-alone" protocol -- it relies on other protocols, such as the Session Initiation Protocol (SIP), to coordinate MRCPv2 clients and servers and manage sessions between them, and the Session Description Protocol (SDP) to describe, discover, and exchange capabilities. It also depends on SIP and SDP to establish the media sessions and associated parameters between the media source or sink and the media server. Once this is done, the MRCPv2 exchange operates over the control session established above, allowing the client to control the media processing resources on the speech resource server. RFC6787 [RFC6787] defines attributes for this purpose.

Name	Notes	Current	Category
resource	Not Impacted	M	NORMAL
channel	Not Impacted	M	NORMAL

RFC6787 Attribute Analysis

5.11. RFC5245 - Interactive Connectivity Establishment (ICE)

RFC5245 [RFC5245] describes a protocol for Network Address Translator(NAT) traversal for UDP-based multimedia sessions established with the offer/answer model. This protocol is called Interactive Connectivity Establishment (ICE). ICE makes use of the Session Traversal Utilities for NAT (STUN) protocol and its extension, Traversal Using Relay NAT (TURN). ICE can be used by any protocol utilizing the offer/answer model, such as the Session Initiation Protocol (SIP).

Name	Notes	Current	Category
ice-lite	Not Impacted	S	NORMAL
ice-options	Not Impacted	S	NORMAL
ice-options	Not Impacted	S	NORMAL
ice-pwd	Per media-level attribute MUST be used per underlying transport flow	B	TRANSPORT
ice-ufrag	Per media-level attribute MUST be used per underlying transport flow	B	TRANSPORT
candidate	Per media-level attribute MUST be used per underlying transport flow		TRANSPORT
remote-candidates	Per media-level attribute MUST be used per underlying transport flow	M	TRANSPORT

RFC5245 Attribute Analysis

5.12. RFC5285 - RTP Header Extensions

RFC5285 [RFC5285] provides a general mechanism to use the header extension feature of RTP (the Real-Time Transport Protocol). It provides the option to use a small number of small extensions in each RTP packet, where the universe of possible extensions is large and registration is de-centralized. The actual extensions in use in a session are signaled in the setup information for that session.

Name	Notes	Current	Category
extmap	Specific RTP extension document MUST be referred	B	SPECIAL

+-----+	+-----+	+-----+	+-----+

RFC5285 Attribute Analysis

5.13. RFC3605 - RTCP attribute in SDP

Originally, SDP assumed that RTP and RTCP were carried on consecutive ports. However, this is not always true when NATs are involved. [RFC3605] specifies an early mechanism to indicate the RTCP port.

Name	Notes	Current	Category
rtcp	Case1:Same RTCP port is repeated across the m=lines. Case2:Different RTCP ports renders multiplexing impossible	M	IDENTICAL

RFC3605 Attribute Analysis

5.14. RFC5576 - Source-Specific SDP Attributes

RFC5576 [RFC5576] defines a mechanism to describe RTP media sources, which are identified by their synchronization source (SSRC) identifiers, in SDP, to associate attributes with these sources, and to express relationships among sources. It also defines several source-level attributes that can be used to describe properties of media sources.

Name	Notes	Current	Category
ssrc	SSRCs repeated over multiple m=lines is forbidden if the m-lines are in the same RTP session.	M	NOT RECOMMENDED
ssrc-group	Refer to section Section 11 for specific analysis of the grouping semantics	M	SPECIAL

cname	Not Impacted [Open Issues: what are the rules for CNAME duplication across sessions?]	SR	NORMAL
previous-ssrc	SSRCs repeated over multiple m=lines complicates multiplexing	SR	NOT RECOMMENDED
fntp	Not Impacted	SR	NORMAL

RFC5576 Attribute Analysis

5.15. RFC6236 - Image Attributes in SDP

RFC6236 [RFC6236] proposes a new generic session setup attribute to make it possible to negotiate different image attributes such as image size. A possible use case is to make it possible for a low-end hand- held terminal to display video without the need to rescale the image, something that may consume large amounts of memory and processing power. The document also helps to maintain an optimal bitrate for video as only the image size that is desired by the receiver is transmitted.

Name	Notes	Current	Category
imageattr	Not Impacted	M	NORMAL

RFC6236 Attribute Analysis

5.16. RFC6285 - Rapid Acquisition of Multicast RTP Sessions

RFC6285 [RFC6285] describes a method using the existing RTP and RTP Control Protocol (RTCP) machinery that reduces the acquisition delay. In this method, an auxiliary unicast RTP session carrying the Reference Information to the receiver precedes or accompanies the multicast stream. This unicast RTP flow can be transmitted at a faster than natural bitrate to further accelerate the acquisition. The motivating use case for this capability is multicast applications that carry real-time compressed audio and video.

Name	Notes	Current	Category
rams-updates	Not recommended	M	NOT RECOMMENDED

RFC6285 Attribute Analysis

5.17. RFC6230 - Media Control Channel Framework

RFC6230 [RFC6230] describes a framework and protocol for application deployment where the application programming logic and media processing are distributed. This implies that application programming logic can seamlessly gain access to appropriate resources that are not co-located on the same physical network entity. The framework uses the Session Initiation Protocol (SIP) to establish an application-level control mechanism between application servers and associated external servers such as media servers.

Name	Notes	Current	Category
cfw-id	Not Applicable	M	NORMAL

RFC6230 Attribute Analysis

5.18. RFC6364 - SDP Elements for FEC Framework

RFC6364 [RFC6364] specifies the use of the Session Description Protocol (SDP) to describe the parameters required to signal the Forward Error Correction (FEC) Framework Configuration Information between the sender(s) and receiver(s). This document also provides examples that show the semantics for grouping multiple source and repair flows together for the applications that simultaneously use multiple instances of the FEC Framework.

Name	Notes	Current	Category
fec-source-flow	Not Impacted	M	NORMAL
fec-repair-flow	Not Impacted	M	NORMAL
repair-window	Not Impacted	M	NORMAL

+-----+	+-----+	+-----+	+-----+	+-----+

RFC6364 Attribute Analysis

5.19. RFC4796 - Content Attribute

RFC4796 [RFC4796] defines a new Session Description Protocol (SDP) media- level attribute, 'content'. The 'content' attribute defines the content of the media stream to a more detailed level than the media description line. The sender of an SDP session description can attach the 'content' attribute to one or more media streams. The receiving application can then treat each media stream differently (e.g., show it on a big or small screen) based on its content.

Name	Notes	Current	Category
content	Not Impacted	M	NORMAL

RFC4796 Attribute Analysis

5.20. RFC3407 - SDP Simple Capability Declaration

RFC3407 [RFC3407] defines a set of Session Description Protocol (SDP) attributes that enables SDP to provide a minimal and backwards compatible capability declaration mechanism.

Name	Notes	Current	Category
sqn	Not Impacted	B	NORMAL
cdsc	Not Impacted	B	NORMAL
cpar	Represents encapsulating attribute	B	TBD
cparmin	Represents encapsulating attribute	B	TBD
cparmax	Represents encapsulating attribute	B	TBD

RFC3407 Attribute Analysis

For attributes that encapsulate other attributes, the negotiation procedures decides the final selection of the attribute from the one or more attributes encapsulated. Multiplexing of media lines with encapsulating attributes requires further analysis.

5.21. RFC6284 - Port Mapping between Unicast and Multicast RTP Sessions

RFC6284 [RFC6284] presents a port mapping solution that allows RTP receivers to choose their own ports for an auxiliary unicast session in RTP applications using both unicast and multicast services. The solution provides protection against denial-of-service or packet amplification attacks that could be used to cause one or more RTP packets to be sent to a victim client

Name	Notes	Current	Category
portmapping-req	Not recommended, if port mapping is required by the application	M	NOT RECOMMENDED

RFC6284 Attribute Analysis

5.22. RFC6714 - MSRP-CEMA

RFC6714 [RFC6714] defines a Message Session Relay Protocol (MSRP) extension, Connection Establishment for Media Anchoring (CEMA). Support of this extension is OPTIONAL. The extension allows middleboxes to anchor the MSRP connection, without the need for middleboxes to modify the MSRP messages; thus, it also enables secure end-to-end MSRP communication in networks where such middleboxes are deployed. This document also defines a Session Description Protocol (SDP) attribute, 'msrp-cema', that MSRP endpoints use to indicate support of the CEMA extension.

Name	Notes	Current	Category
msrp-cema	Not Impacted	M	NORMAL

RFC6714 Attribute Analysis

5.23. RFC4583 - SDP Format for BFCP Streams

RFC4583 [RFC4583] document specifies how to describe Binary Floor Control Protocol (BFCP) streams in Session Description Protocol (SDP) descriptions. User agents using the offer/answer model to establish BFCP streams use this format in their offers and answers

Name	Notes	Current	Category
floorctrl	Not Impacted	M	NORMAL
confid	Not Impacted	M	NORMAL
userid	Not Impacted	M	NORMAL
floorid	Not Impacted	M	NORMAL

RFC4583 Attribute Analysis

5.24. RFC5547 - SDP Offer/Answer for File Transfer

RFC5547 [RFC5547] provides a mechanism to negotiate the transfer of one or more files between two endpoints by using the Session Description Protocol (SDP) offer/answer model specified in [RFC3264].

Name	Notes	Current	Category
file-selector	Not Impacted	M	NORMAL
file-transfer-id	Not Impacted	M	NORMAL
file-disposition	Not Impacted	M	NORMAL
file-date,file-iconfile-range	Not Impacted	M	NORMAL
file-iconfile-range	Not Impacted	M	NORMAL
file-iconfile-range	Not Impacted	M	NORMAL

+-----+	+-----+	+-----+	+-----+

RFC5547 Attribute Analysis

5.25. draft-ietf-mmusic-media-loopback

[MEDIA_LOOPBACK] adds new SDP media types and attributes, which enable establishment of media sessions where the media is looped back to the transmitter. Such media sessions will serve as monitoring and troubleshooting tools by providing the means for measurement of more advanced VoIP, Real-time Text and Video over IP performance metrics.

Name	Notes	Current	Category
loopback rtp-pkt-loopback	The attribute MUST be repeated across all m=lines multiplexed	M	IDENTICAL
loopback rtp-media-loopback	Not Impacted	M	NORMAL
loopback-source	Not Impacted	M	NORMAL
loopback-mirror	Not Impacted	M	NORMAL

draft-ietf-mmusic-media-loopback Attribute Analysis

5.26. RFC5760 - RTCP with Unicast Feedback

RFC5760 [RFC5760] specifies an extension to the Real-time Transport Control Protocol (RTCP) to use unicast feedback to a multicast sender. The proposed extension is useful for single-source multicast sessions such as Source-Specific Multicast (SSM) communication where the traditional model of many-to-many group communication is either not available or not desired.

Name	Notes	Current	Category
rtcp-unicast	The attribute MUST be reported across all m=lines multiplexed	M	IDENTICAL

+-----+	+-----+	+-----+	+-----+

RFC5760 Attribute Analysis

5.27. RFC3611 - RTCP XR

RFC3611 [RFC3611] defines the Extended Report (XR) packet type for the RTP Control Protocol (RTCP), and defines how the use of XR packets can be signaled by an application if it employs the Session Description Protocol (SDP).

Name	Notes	Current	Category
rtcp-xr	The attribute MUST be reported across all m=lines multiplexed	B	IDENTICAL

RFC3611 Attribute Analysis

5.28. RFC5939 - SDP Capability Negotiation

RFC5939 [RFC5939] defines a general SDP Capability Negotiation framework. It also specifies how to provide attributes and transport protocols as capabilities and negotiate them using the framework. Extensions for other types of capabilities (e.g., media types and media formats) may be provided in other documents.

Name	Notes	Current	Category
pcfg	Depends on capability being negotiated	M	SPECIAL
acfg	Depends on capability being negotiated	M	SPECIAL
csup	Not Impacted	B	NORMAL
creq	Not Impacted	B	NORMAL
acap	Represents encapsulation attribute	B	TBD
tcap	Represents encapsulation attribute	B	TBD

RFC5939 Attribute Analysis

For attributes that encapsulate other attributes, the negotiation procedures decides the final selection of the attribute from the one or more attributes encapsulated. Multiplexing of media lines with encapsulating attributes requires further analysis.

5.29. RFC6871 - SDP Media Capabilities Negotiation

Session Description Protocol (SDP) capability negotiation provides a general framework for indicating and negotiating capabilities in SDP. The base framework defines only capabilities for negotiating transport protocols and attributes. [MEDIA_CAP] extends the framework by defining media capabilities that can be used to negotiate media types and their associated parameters.

Name	Notes	Current	Category
rmcap	Not Impacted	B	NORMAL
omcap	Not Impacted	B	NORMAL
mfcap	Not Impacted	B	NORMAL
mscap	mscap represents encapsulating attribute	B	TBD
lcfg	Need to revisit this later	B	TBD
sescap	Not Impacted	S	NORMAL

draft-ietf-mmusic-sdp-media-capabilities Attribute Analysis

For attributes that encapsulate other attributes, the negotiation procedures decides the final selection of the attribute from the one or more attributes encapsulated. Multiplexing of media lines with encapsulating attributes requires further analysis.

5.30. RFC4567 - Key Management Extensions for SDP and RTSP

RFC4567 [RFC4567] defines general extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP) to carry messages, as specified by a key management protocol, in order to secure the media. These extensions are presented as a framework, to be used by one or more key management protocols. As such, their use

is meaningful only when complemented by an appropriate key management protocol.

Name	Notes	Current	Category
key-mgmt	Key management protocol MUST be identical across all the m=lines	B	IDENTICAL

RFC4567 Attribute Analysis

5.31. RFC4572 - Comedia over TLS in SDP

RFC4572 [RFC4572] specifies how to establish secure connection-oriented media transport sessions over the Transport Layer Security (TLS) protocol using the Session Description Protocol (SDP). It defines a new SDP protocol identifier, 'TCP/TLS'. It also defines the syntax and semantics for an SDP 'fingerprint' attribute that identifies the certificate that will be presented for the TLS session. This mechanism allows media transport over TLS connections to be established securely, so long as the integrity of session descriptions is assured.

Name	Notes	Current	Category
fingerprint	Fingerprint value MUST be identical across all the m=lines	B	IDENTICAL

RFC4572 Attribute Analysis

5.32. RFC4570 - SDP Source Filters

RFC4570 [RFC4570] describes how to adapt the Session Description Protocol (SDP) to express one or more source addresses as a source filter for one or more destination "connection" addresses. It defines the syntax and semantics for an SDP "source-filter" attribute that may reference either IPv4 or IPv6 address(es) as either an inclusive or exclusive source list for either multicast or unicast destinations. In particular, an inclusive source-filter can be used to specify a Source-Specific Multicast (SSM) session

Name	Notes	Current	Category
source-filter	he attribute MUST be repeated across all m=lines multiplexed	B	IDENTICAL

RFC4570 Attribute Analysis

5.33. RFC6128 - RTCP Port for Multicast Sessions

The Session Description Protocol (SDP) has an attribute that allows RTP applications to specify an address and a port associated with the RTP Control Protocol (RTCP) traffic. In RTP-based source-specific multicast (SSM) sessions, the same attribute is used to designate the address and the RTCP port of the Feedback Target in the SDP description. However, the RTCP port associated with the SSM session itself cannot be specified by the same attribute to avoid ambiguity, and thus, is required to be derived from the "m=" line of the media description. Deriving the RTCP port from the "m=" line imposes an unnecessary restriction. RFC6128 [RFC6128] removes this restriction by introducing a new SDP attribute.

Name	Notes	Current	Category
multicast-rtcp	Multicast RTCP port MUST be identical across all the m=lines	B	IDENTICAL

RFC6128 Attribute Analysis

5.34. RFC6189 - ZRTP

RFC6189 [RFC6189] defines ZRTP, a protocol for media path Diffie-Hellman exchange to agree on a session key and parameters for establishing unicast Secure Real-time Transport Protocol (SRTP) sessions for Voice over IP (VoIP) applications.

Name	Notes	Current	Category
zrtp-hash	Complicates if all the m=lines are not authenticated as given in the example below	M	NOT RECOMMENDED

RFC6189 Attribute Analysis

Example: Multiplexing media descriptions having attribute zrtp-hash defined with the media descriptions lacking it, would either complicate the handling of multiplexed stream or fail multiplexing.

```

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=
c=IN IP4 client.biloxi.example.com
t=0 0
m=audio 3456 RTP/AVP 97
a=rtpmap:97 iLBC/8000
<allOneLine>
a=zrtp-hash:1.10 fe30efd02423cb054e50efd0248742ac7a52c8f91bc2
df881ae642c371ba46df
</allOneLine>
m=video 34567 RTP/AVP 31
a=rtpmap:31 H261/90000

```

5.35. RFC4145 - Connection-Oriented Media

RFC4145 [RFC4145] describes how to express media transport over TCP using the Session Description Protocol (SDP). It defines the SDP 'TCP' protocol identifier, the SDP 'setup' attribute, which describes the connection setup procedure, and the SDP 'connection' attribute, which handles connection reestablishment.

Name	Notes	Current	Category
setup	Should be identical across all m=lines	B	R
connection	Should be identical across all m=lines	B	R

RFC4145 Attribute Analysis

5.36. RFC5159 - OMA BCAST SDP Attributes

RFC5159 [RFC5159] provides descriptions of Session Description Protocol (SDP) attributes used by the Open Mobile Alliance's Broadcast Service and Content Protection specification.

Name	Notes	Current	Category
bcastversion	Might cause legacy interop issues	S	TBD
stkmstream	Might cause legacy interop purposes	B	TBD
SRTPAuthentication	Might cause legacy interop issues	M	TBD
SRTPROCTxRate	Might cause legacy interop issues	M	TBD

RFC5159 Attribute Analysis

5.37. RFC6193 - Media Description for IKE in SDP

RFC6193 [RFC6193] specifies how to establish a media session that represents a virtual private network using the Session Initiation Protocol for the purpose of on-demand media/application sharing between peers. It extends the protocol identifier of the Session Description Protocol (SDP) so that it can negotiate use of the Internet Key Exchange Protocol (IKE) for media sessions in the SDP offer/answer model.

Name	Notes	Current	Category
ike-setup	Attribute MUST be identical across all the m=lines	B	IDENTICAL
psk-fingerprint	Attribute MUST be identical across all the m=lines	B	IDENTICAL

ike-esp	Attribute MUST be identical across all the m=lines	B	IDENTICAL
ike-esp-udpencap	Attribute MUST be identical across all the m=lines	B	IDENTICAL

RFC6193 Attribute Analysis

With the above SDP constraints, a session multiplexed with multiple m=lines will use only one IPSec association for all of the m= lines.

5.38. RFC6064 - SDP and RTSP Extensions for 3GPP

The Packet-switched Streaming Service (PSS) and the Multimedia Broadcast/Multicast Service (MBMS) defined by 3GPP use the Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP) with some extensions. RFC6064 [RFC6064] provides information about these extensions and registers the RTSP and SDP extensions with IANA.

Name	Notes	Current	Category
X-predecbufsize	Case1:Aggregate total when video m-lines are muxed Case2:Multiplexing with audio m=lines is invalid	M	NOT RECOMMENDED
X-initpredecbufperiod	Case1:Aggregate total when video m-lines are muxed Case2:Multiplexing with audio m=lines is invalid	M	NOT RECOMMENDED
X-initpostdecbufperiod	Case1:Aggregate total when video m-lines are muxed Case2:Multiplexing with audio m=lines is invalid	M	NOT RECOMMENDED
X-decbyterate	Case1:Aggregate total when video m-lines are muxed Case2:Multiplexing with audio m=lines is invalid	M	NOT RECOMMENDED
3gpp-videopostdecbufsize	Case1:Aggregate total when video m-lines are muxed. Case2:Multiplexing with audio m=lines is invalid	M	NOT RECOMMENDED
framesize	Not Impacted	M	NORMAL
3GPP-Integrity-Key	Not Impacted	S	NORMAL
3GPP-SRTP-Config	Same config SHALL apply to all the m=lines multiplexed	M	NORMAL

alt,alt-default-id	Specifying alternate m=lines when session with multiple m=lines of different types cannot be clearly specified	M	TBD
alt-group	Complicates selection of alternate m=lines grouped with alt-group on multiplexing	M	TBD
3GPP-Adaptation-Support	Not recommended for legacy interop purposes	M	TBD
3GPP-QoE-Metricsn	Not recommended for legacy interop purposes	B	TBD
3GPP-Asset-Information	Not recommended for legacy interop purposes	B	TBD
mbms-mode	Not recommended for legacy interop purposes	B	TBD
mbms-flowid	Multiplexing multiple m=lines complicates FEC mappings to the transport addresses.	M	TBD
mbms-repair	Not recommended for legacy interop purposes	B	TBD

RFC6064 Attribute Analysis

5.39. RFC3108 - ATM SDP

RFC3108 [RFC3108] describes conventions for using the Session Description Protocol (SDP) for characterizing ATM bearer connections using an AAL1, AAL2 or AAL5 adaptation layers.

For AAL1, AAL2 and AAL5, bearer connections can be used to transport single media streams. In addition, for AAL1 and AAL2, multiple media streams may be multiplexed into a bearer connection. For all adaptation types (AAL1, AAL2 and AAL5), bearer connections may be bundled into a single media group. In all cases addressed by RFC3108, a real-time media stream (voice, video, voiceband data, pseudo-wire and others) or a multiplex of media streams is mapped directly into an ATM connection. RFC3108 does not address cases where ATM serves as a low-level transport pipe for IP packets which in turn may carry one or more real-time (e.g. VoIP) media sessions with a life-cycle different from that of the underlying ATM transport.

Name	Notes	Current	Category
aalType	Not Impacted	B	NORMAL
eecid	Not Impacted	B	NORMAL
aalType	Not Impacted	B	NORMAL
capability	Not Impacted	B	NORMAL
qosClass	Not Impacted	B	NORMAL
bcob	Not Impacted	B	NORMAL
stc	Not Impacted	B	NORMAL
upcc	Not Impacted	B	NORMAL
atmQOSparms	Not Impacted	B	NORMAL
atmTrfcDesc	Not Impacted	B	NORMAL
abrParms	Not Impacted	B	NORMAL
abrSetup	Not Impacted	B	NORMAL
bearerType	Not Impacted	B	NORMAL
lij	Not Impacted	B	NORMAL
anycast	Not Impacted	B	NORMAL
cache	Not Impacted	B	NORMAL
bearerSigIE	Not Impacted	B	NORMAL
aalApp	Not Impacted	B	NORMAL
cbrRate	Not Impacted	B	NORMAL
sbc	Not Impacted	B	NORMAL
clkrec	Not Impacted	B	NORMAL
fec	Not Impacted	B	NORMAL
prtfl	Not Impacted	B	NORMAL
structure	Not Impacted	B	NORMAL
cpsSDUsize	Not Impacted	B	NORMAL
aal2CPS	Not Impacted	B	NORMAL

aal2CPSSDUrate	Not Impacted	B	NORMAL
aal2sscs3661unassured	Not Impacted	B	NORMAL
aal2sscs3661assured	Not Impacted	B	NORMAL
aal2sscs3662	Not Impacted	B	NORMAL
aal5sscscop	Not Impacted	B	NORMAL
atmmmap	Not Impacted	B	NORMAL
silenceSupp	Not Impacted	B	NORMAL
ecan	Not Impacted	B	NORMAL
gc	Not Impacted	B	NORMAL
profileDesc	Not Impacted	B	NORMAL
vsel	Not Impacted	B	NORMAL
dsel	Not Impacted	B	NORMAL
fsel	Not Impacted	B	NORMAL
onewaySel	Not Impacted	B	NORMAL
codeconfig	Not Impacted	B	NORMAL
isup_usi	Not Impacted	B	NORMAL
isup_usi	Not Impacted	B	NORMAL
chain	Not Impacted	B	NORMAL

RFC3108 Attribute Analysis

5.40. 3GPP TS 24.182

3GPP TS 24.182 [3GPP TS 24.182] specifies IP multimedia subsystem Custom Alerting tones

Name	Notes	Current	Category
g.3gpp.cat	Usage defined for the IP Multimedia Subsystem	M	NORMAL

3GPP TS 24.182 Attribute Analysis

5.41. 3GPP TS 24.183

3GPP TS 24.183 [3GPP TS 24.183] specifies IP multimedia subsystem Custom Ringing Signal

Name	Notes	Current	Category
g.3gpp.crs	Usage defined for the IP Multimedia Subsystem	M	NORMAL

+-----+	+-----+	+-----+	+-----+

3GPP TS 24.183 Attribute Analysis

5.42. 3GPP TS 24.229

3GPP TS 24.229 [3GPP TS 24.229] IP multimedia call control protocol based on Session Initial protocol and Session Description Protocol.

Name	Notes	Current	Category
secondary-realm	Per media-level attribute MUST be used per underlying transport	M	TRANSPORT
visited-realm	Per media-level attribute MUST be used per underlying transport	M	TRANSPORT
omr-m-cksum	Not Impacted	M	NORMAL
omr-s-cksum	Not Impacted	M	NORMAL
omr-m-att	Not Impacted	M	NORMAL
omr-s-bw	Not Impacted	M	NORMAL
omr-s-bw	Not Impacted	M	NORMAL
omr-m-att	Not Impacted	M	NORMAL
omr-codecs	Not Impacted	M	NORMAL

3GPP TS 24.229 Attribute Analysis

5.43. ITU T.38

ITU T.38[T.38] defines procedures for real-time Group 3 facsimile communications over IP networks.

Name	Notes	Current	Category
T38FaxVersion	Not Impacted	S	NORMAL
T38MaxBitRate	Not Impacted	S	NORMAL
T38FaxFillBitRemoval	Not Impacted	S	NORMAL
T38FaxTranscodingMMR	Not Impacted	S	NORMAL
T38FaxTranscodingJBIG	Not Impacted	S	NORMAL
T38FaxRateManagement	Not Impacted	S	NORMAL
T38FaxMaxBuffer	Not Impacted	S	NORMAL
T38FaxMaxDatagram	Not Impacted	S	NORMAL
T38FaxUdpEC	Not Impacted	S	NORMAL

Historic Attribute Analysis

The ITU T.38 attributes are clearly unaffected by multiplexing and are specific to the working of the fax protocol itself.

5.44. ITU-T H.248.15

ITU-T H.248.15 [H.248.15] defines Gateway Control Protocol SDP H.248 package attribute

Name	Notes	Current	Category
h248item	It is also only applicable for signaling the inclusion of H.248 extension packages to a gateway via the local and remote descriptors. The attribute itself is unaffected by multiplexing, but the packaged referenced in a specific use of the attribute may be impacted. Further analysis of each package is needed to determine if there is an issue. This is only a concern in environments using a decomposed server/gateway with H.248 signaled between them. The ITU-T will need to do further analysis of various packages when they specify how to signal the use of multiplexing to a gateway.	B	SPECIAL

Historic Attribute Analysis

5.45. RFC4975 - The Message Session Relay Protocol

RFC4975 [RFC4975] the Message Session Relay Protocol, a protocol for transmitting a series of related instant messages in the context of a session. Message sessions are treated like any other media stream when set up via a rendezvous or session creation protocol such as the Session Initiation Protocol.

Name	Notes	Current	Category
accept-types	Not Impacted	M	NORMAL
accept-wrapped-types	Not Impacted	M	NORMAL
max-size	Not Impacted	M	NORMAL
path	Not Impacted	M	NORMAL

RFC4975 Attribute Analysis

5.46. Historical

This section specifies analysis for the attributes that are included for historic usage alone by the [IANA].

Name	Notes	Current	Category
rtppred1	Not Applicable	Not-Applicable	TBD
rtppred2	Not Applicable	Not-Applicable	TBD
PSCid	Not Applicable	Not-Applicable	TBD
bc_service	Not Applicable	Not-Applicable	TBD
bc_program	Not Applicable	Not-Applicable	TBD
bc_service_package	Not Applicable	Not-Applicable	TBD

Unknowns Attribute Analysis

6. bwtype Attribute Analysis

This section specifies handling of specific bandwidth attributes when used in multiplexing scenarios.

6.1. RFC4566 - SDP: Session Description Protocol

Name	Notes	Current	Category
bwtype:CT	Aggregate bandwidth for the conference	S	NORMAL

bwtype:AS	There are 2 interpretations for this attribute As a session attribute, it specifies the session aggregate unless media-level b=RR and/or b=RS attributes are used. Under this interpretation the multiplexing scheme has no impact and belongs to NORMAL category. For the media level usage, the aggregate of individual bandwidth values is considered.	B	NORMAL, SUM
-----------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---	-------------

RFC4566 bwtype Analysis

6.2. RFC3556 - SDP Bandwidth Modifiers for RTCP Bandwidth

RFC3556 [RFC3556] defines an extension to the Session Description Protocol (SDP) to specify two additional modifiers for the bandwidth attribute. These modifiers may be used to specify the bandwidth allowed for RTP Control Protocol (RTCP) packets in a Real-time Transport Protocol (RTP) session

Name	Notes	Current	Category
bwtype:RS	Session level usage represents session aggregate and media level usage indicates SUM of the individual values while multiplexing	B	NORMAL, SUM
bwtype:RR	Session level usage represents session aggregate and media level usage indicates SUM of the individual values while multiplexing	B	NORMAL, SUM

RFC3556 bwtype Analysis

6.3. RFC3890 - Bandwidth Modifier for SDP

RFC3890 [RFC3890] defines a Session Description Protocol (SDP) Transport Independent Application Specific Maximum (TIAS) bandwidth modifier that does not include transport overhead; instead an additional packet rate attribute is defined. The transport independent bit-rate value together with the maximum packet rate can then be used to calculate the real bit-rate over the transport actually used.

Name	Notes	Current	Category
bwtype:TIAS	The usage of TIAS is not clearly defined Offer/Answer usage.	B	TBD
maxprate	The usage of TIAS and maxprate is not well defined under multiplexing	B	TBD

RFC3890 bwtype Analysis

The intention of TIAS is that the media level bit-rate is multiplied with the known per-packet overhead for the selected transport and the maxprate value to determine the worst case bit-rate from the transport to more accurately capture the required usage. Summing TIAS values independently across m=lines and multiplying the computed sum with maxprate and the per-packet overhead would inflate the value significantly. Instead performing multiplication and adding the individual values is a more appropriate usage. This still ignores the fact that this is a send side declaration, and not intended for receiver negotiation.

7. rtcp-fb Attribute Analysis

This section analyzes rtcp-fb SDP attributes [RTCP-FB].

7.1. RFC4585 - RTP/AVPF

RFC4585 [RFC4585] defines an extension to the Audio-visual Profile (AVP) that enables receivers to provide, statistically, more immediate feedback to the senders and thus allows for short-term adaptation and efficient feedback-based repair mechanisms to be implemented.

Attr Name	Notes	Current	Category
ack	Not Impacted	M	NORMAL
app	Not Impacted	M	NORMAL
nack	Not Impacted	M	NORMAL
trr-int	Not Impacted	M	NORMAL

RFC4585 Attribute Analysis

7.2. RFC5104 - Codec Control Messages in AVPF

RFC5104 [RFC5104] specifies a few extensions to the messages defined in the Audio-Visual Profile with Feedback (AVPF). They are helpful primarily in conversational multimedia scenarios where centralized multipoint functionalities are in use. However, some are also usable in smaller multicast environments and point-to-point calls.

Attr Name	Notes	Current	Category
ccm	Not Impacted	M	Normal

RFC5104 Attribute Analysis

8. rtcp-fb "ack/nack" Attribute Analysis

This section analyzes rtcp-fb SDP attributes specific to ack and nack feedback types [ACK-NACK].

8.1. RFC4585 - RTP/AVPF

RFC4585 [RFC4585] defines an extension to the Audio-visual Profile (AVP) that enables receivers to provide, statistically, more immediate feedback to the senders and thus allows for short-term adaptation and efficient feedback-based repair mechanisms to be implemented.

Attr Name	Notes	Current	Category
nack sli	Not Impacted	M	NORMAL
nack pli	Not Impacted	M	NORMAL
ack rpsi	Not Impacted	M	NORMAL
ack app	Feedback parameters MUST be handled in the app specific way when multiplexed	M	SPECIAL
nack rpsi	Not Impacted	M	NORMAL
nack app	Feedback parameters MUST be handled in the app specific way when multiplexed	M	SPECIAL

RFC4585 Attribute Analysis

8.2. RFC6285 - Unicast-Based RAMS

Name	Notes	Current	Category
nack rai	Not Impacted	M	NORMAL

RFC6285 Attribute Analysis

8.3. RFC6679 - ECN for RTP over UDP/IP

RFC6679 [RFC6679] specifies how Explicit Congestion Notification (ECN) can be used with the Real-time Transport Protocol (RTP) running over UDP, using the RTP Control Protocol (RTCP) as a feedback mechanism. It defines a new RTCP Extended Report (XR) block for periodic ECN feedback, a new RTCP transport feedback message for timely reporting of congestion events, and a Session Traversal Utilities for NAT (STUN) extension used in the optional

initialisation method using Interactive Connectivity Establishment (ICE)

Name	Notes	Current	Category
ecn-capable-rtp	ECN markup are enabled at the RTP Session level	M	IDENTICAL
nack ecn	This attribute enables ECN at the RTP session level	M	IDENTICAL

RFC6679 Attribute Analysis

8.4. RFC6642 - Third-Party Loss Report

In a large RTP session using the RTP Control Protocol (RTCP) feedback mechanism defined in RFC 4585 [RFC4585], a feedback target may experience transient overload if some event causes a large number of receivers to send feedback at once. This overload is usually avoided by ensuring that feedback reports are forwarded to all receivers, allowing them to avoid sending duplicate feedback reports. However, there are cases where it is not recommended to forward feedback reports, and this may allow feedback implosion. RFC6642 [RFC6642] memo discusses these cases and defines a new RTCP Third-Party Loss Report that can be used to inform receivers that the feedback target is aware of some loss event, allowing them to suppress feedback. Associated Session Description Protocol (SDP) signaling is also defined.

Name	Notes	Current	Category
tllei	Not Impacted	M	NORMAL
pslei	Not Impacted	M	NORMAL

RFC6642 Attribute Analysis

9. Codec Control Messages Analysis

This section analyzes rtcp-fb Codec Control Message [CCM].

9.1. RFC5104 - Codec Control Messages in AVPF

Attr Name	Notes	Current	Category
fir	Not Impacted	M	NORMAL
tmmbr	Not Impacted	M	NORMAL
tstr	Not Impacted	M	NORMAL
vbcm	Not Impacted	M	NORMAL

RFC5104 Attribute Analysis

10. group Attribute Analysis

This section analyzes SDP "group" semantics [GROUP-SEM].

10.1. RFC5888 - SDP Grouping Framework

RFC5888 [RFC5888] defines a framework to group "m" lines in the Session Description Protocol (SDP) for different purposes.

Name	Notes	Current	Category
group:LS	Not Impacted	S	NORMAL
group:FID	Not Impacted	S	NORMAL

RFC5888 Attribute Analysis

10.2. RFC3524 - Mapping Media Streams to Resource Reservation Flows

RFC3524 [RFC3524] defines an extension to the Session Description Protocol (SDP) grouping framework. It allows requesting a group of media streams to be mapped into a single resource reservation flow. The SDP syntax needed is defined, as well as a new "semantics"

attribute called Single Reservation Flow (SRF).

Name	Notes	Current	Category
group:SRF	Not Impacted	S	NORMAL

RFC3524 Attribute Analysis

10.3. RFC4091 - ANAT Semantics

RFC4091 [RFC4091] defines the Alternative Network Address Types (ANAT) semantics for the Session Description Protocol (SDP) grouping framework. The ANAT semantics allow alternative types of network addresses to establish a particular media stream.

Name	Notes	Current	Category
group:ANAT	Not Impacted	S	NOT RECOMMENDED

RFC4091 Attribute Analysis

10.4. RFC5956 - FEC Grouping Semantics in SDP

RFC5956 [RFC5956] defines the semantics for grouping the associated source and FEC-based (Forward Error Correction) repair flows in the Session Description Protocol (SDP). The semantics defined in the document are to be used with the SDP Grouping Framework (RFC 5888). These semantics allow the description of grouping relationships between the source and repair flows when one or more source and/or repair flows are associated in the same group, and they provide support for additive repair flows. SSRC-level (Synchronization Source) grouping semantics are also defined in this document for Real-time Transport Protocol (RTP) streams using SSRC multiplexing.

Name	Notes	Current	Category
group:FEC-FR	Not Impacted	S	NORMAL

RFC5956 Attribute Analysis

10.5. RFC5583 - Signaling Media Decoding Dependency in SDP

RFC5583 [RFC5583] defines semantics that allow for signaling the decoding dependency of different media descriptions with the same media type in the Session Description Protocol (SDP). This is required, for example, if media data is separated and transported in different network streams as a result of the use of a layered or multiple descriptive media coding process.

Name	Notes	Current	Category
depend lay	Not Impacted	M	NORMAL
depend mdc	Not Impacted	M	NORMAL

RFC5583 Attribute Analysis

11. ssrc-group Attribute Analysis

This section analyzes "ssrc-group" semantics [SSRC-GROUP].

11.1. RFC5576 - Source-Specific SDP Attributes

Name	Notes	Current	Category
FID	Not Impacted	M	NORMAL
FEC	Not Impacted	M	NORMAL
FEC-FR	Not Impacted	M	NORMAL

RFC5576 Attribute Analysis

12. QoS Mechanism Token Analysis

This section analyzes QoS tokens specified with SDP[QOS].

12.1. RFC5432 - QoS Mechanism Selection in SDP

Name	Notes	Current	Category
rsvp	Not Impacted, since QoS mechanisms are applied per flow.	B	NORMAL
nsis	Not Impacted, since QoS mechanisms are applied per flow.	B	NORMAL

RFC5432 Attribute Analysis

13. k= Attribute Analysis

13.1. RFC4566 SDP: Session Description Protocol

Name	Notes	Current	Category
k=	It is NOT recommended to use this attribute	S	NOT RECOMMENDED

RFC4566 Attribute Analysis

14. content Attribute Analysis

14.1. RFC4796 - MSRP Relays

Name	Notes	Current	Category
content:slides	Not Impacted	M	NORMAL
content:speaker	Not Impacted	M	NORMAL
content:main	Not Impacted	M	NORMAL
content:sl	Not Impacted	M	NORMAL
content:alt	Not Impacted	M	NORMAL

RFC4796 Attribute Analysis

15. Payload Formats

15.1. RFC5109 - RTP Payload Format for Generic FEC

RFC5109 [RFC5109] describes a payload format for generic Forward Error Correction (FEC) for media data encapsulated in RTP. It is based on the exclusive-or (parity) operation. The payload format allows end systems to apply protection using various protection lengths and levels, in addition to using various protection group sizes to adapt to different media and channel characteristics. It enables complete recovery of the protected packets or partial recovery of the critical parts of the payload depending on the packet loss situation.

Name	Notes	Current	Category
audio/ulpfec	Not recommended for multiplexing due to reuse of SSRCs	M	NOT RECOMMENDED
video/ulpfec	Not recommended for multiplexing due to reuse of SSRCs	M	NOT RECOMMENDED

text/ulpfec	Not recommended for multiplexing due to reuse of SSRCs	M	NOT RECOMMENDED
application/ulpfec	Not recommended for multiplexing due to reuse of SSRCs	M	NOT RECOMMENDED

RFC5109 Payload Format Analysis

Draft draft-lennox-payload-ulp-ssrc-mux proposes a simple fix to make it possible to use ULP with multiplexing and ULP is allowed when used with that.

16. IANA Considerations

IANA shall register categories from this specification by expanding the Session Description Protocol (SDP) Parameters table with a column listing categories against each SDP parameter.

Category
NORMAL
NOT RECOMMENDED
IDENTICAL
TRANSPORT
SPECIAL

17. Security Considerations

All the attributes which involve security key needs a careful review to ensure two-time pad vulnerability is not created

18. Acknowledgments

I would like to thank Cullen Jennings for suggesting the categories, contributing text and helping review the draft.

I would like also to thank following experts on their inputs and reviews as listed - Rohan Mahy(5.45), Eric Burger(5.22), Christian Huitema(5.13), Christer Holmberg(5.22,5.40,5.41), Richard Ejzak (5.42,5.43,5.44), Colin Perkins(5.7,5.8), Magnus westerlund(5.3,5.9,6.1,6.2,6.3,8.3), Subha Dhesikan(5.5,12.1), Dan Wing(5.6,5.34,5.37), Eric Rescorla (5.11), Ali C Begen(5.1,5.16,5.18,8.2,5.33,13.1), Rajesh Kumar(5.39), Flemming Andreassen(5.28,5.29,5.20)

19. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-nandakumar-mmusic-mux-attributes-02

- o Updated Sections 5.3,5.5,5.6,5.7,5.9,5.8,5.11,5.13,5.20,5.22,5.28, 5.29,5.34,5.37,5.39,5.40, 5.41,5.42,5.43,5.44,5.45,6.1,6.2,6.3,8,3,12.1 based on the inputs from the respective RFC Authors.

Changes from draft-nandakumar-mmusic-mux-attributes-01

- o Replaced Category BAD with NOT RECOMMENDED.
- o Added Category TBD.
- o Updated IANA Consideration Section.

Changes from draft-nandakumar-mmusic-mux-attributes-00

- o Added new section for dealing with FEC payload types.

20. References

20.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

20.2. Informative References

- [3GPP TS 24.182]
"IP Multimedia Subsystem (IMS) Customized Alerting Tones

- (CAT); Protocol specification",
<<http://www.3gpp.org/ftp/Specs/html-info/24182.htm>>.
- [3GPP TS 24.183]
"IP Multimedia Subsystem (IMS) Customized Ringing Signal (CRS); Protocol specification",
<<http://www.3gpp.org/ftp/Specs/html-info/24183.htm>>.
- [3GPP TS 24.229]
"IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP);",
<<http://www.3gpp.org/ftp/Specs/html-info/24229.htm>>.
- [ACK-NACK]
"Session Description Protocol (SDP) RTCP ACK/NACK Feedback attributes", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-15>>.
- [CCM]
"Session Description Protocol (SDP) RTCP-FB Codec Control Messages", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-19>>.
- [GROUP-SEM]
"Session Description Protocol (SDP) "group" semantics", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-13>>.
- [H.248.15]
"Gateway control protocol: SDP H.248 package attribute",
<<http://www.itu.int/rec/T-REC-H.248.15>>.
- [I-D.ietf-avt-multiplexing-rtp]
El-Khatib, K., Luo, G., Bochmann, G., and Pinjiang. Feng,
"Multiplexing Scheme for RTP Flows between Access Routers", Internet-Draft <http://tools.ietf.org/html/draft-ietf-avt-multiplexing-rtp-01>, October 1999.
- [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.
- [IANA]
"Session Description Protocol (SDP) Parameters", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml>>.

- [MEDIA_CAP] Kaplan, H., Hedayat, K., and N. Venna, "Session Description Protocol (SDP) Media Capabilities Negotiation", draft-ietf-mmusic-sdp-media-capabilities-17 (work in progress), January 2013.
- [MEDIA_LOOPBACK] Kaplan, H., Hedayat, K., Venna, N., Jones, P., and N. Stratton, "An Extension to the Session Description Protocol (SDP) and Real-time Transport Protocol (RTP) for Media Loopback", draft-ietf-mmusic-media-loopback-27 (work in progress), January 2013.
- [QOS] "Session Description Protocol (SDP) QoS Mechanism Tokens", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-20>>.
- [RFC3108] Kumar, R. and M. Mostafa, "Conventions for the use of the Session Description Protocol (SDP) for ATM Bearer Connections", RFC 3108, May 2001.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3407] Andreasen, F., "Session Description Protocol (SDP) Simple Capability Declaration", RFC 3407, October 2002.
- [RFC3524] Camarillo, G. and A. Monrad, "Mapping of Media Streams to Resource Reservation Flows", RFC 3524, April 2003.
- [RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.
- [RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC3890] Westerlund, M., "A Transport Independent Bandwidth Modifier for the Session Description Protocol (SDP)", RFC 3890, September 2004.
- [RFC4091] Camarillo, G. and J. Rosenberg, "The Alternative Network

Address Types (ANAT) Semantics for the Session Description Protocol (SDP) Grouping Framework", RFC 4091, June 2005.

- [RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.
- [RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.
- [RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4570] Quinn, B. and R. Finlayson, "Session Description Protocol (SDP) Source Filters", RFC 4570, July 2006.
- [RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, July 2006.
- [RFC4574] Levin, O. and G. Camarillo, "The Session Description Protocol (SDP) Label Attribute", RFC 4574, August 2006.
- [RFC4583] Camarillo, G., "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", RFC 4583, November 2006.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.
- [RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

- [RFC5159] Dondeti, L. and A. Jerichow, "Session Description Protocol (SDP) Attributes for Open Mobile Alliance (OMA) Broadcast (BCAST) Service and Content Protection", RFC 5159, March 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, July 2006.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5432] Polk, J., Dhesikan, S., and G. Camarillo, "Quality of Service (QoS) Mechanism Selection in the Session Description Protocol (SDP)", RFC 5432, March 2009.
- [RFC5506] Johansson, I., "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.
- [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.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5762] Perkins, C., "RTP and the Datagram Congestion Control Protocol (DCCP)", RFC 5762, April 2010.
- [RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.

- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, September 2010.
- [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.
- [RFC6064] Westerlund, M. and P. Frojdh, "SDP and RTSP Extensions Defined for 3GPP Packet-Switched Streaming Service and Multimedia Broadcast/Multicast Service", RFC 6064, January 2011.
- [RFC6128] Begen, A., "RTP Control Protocol (RTCP) Port for Source-Specific Multicast (SSM) Sessions", RFC 6128, February 2011.
- [RFC6189] Zimmermann, P., Johnston, A., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, April 2011.
- [RFC6193] Saito, M., Wing, D., and M. Toyama, "Media Description for the Internet Key Exchange Protocol (IKE) in the Session Description Protocol (SDP)", RFC 6193, April 2011.
- [RFC6230] Boulton, C., Melanchuk, T., and S. McGlashan, "Media Control Channel Framework", RFC 6230, May 2011.
- [RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.
- [RFC6284] Begen, A., Wing, D., and T. Van Caenegem, "Port Mapping between Unicast and Multicast RTP Sessions", RFC 6284, June 2011.
- [RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.
- [RFC6364] Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, October 2011.
- [RFC6642] Wu, Q., Xia, F., and R. Even, "RTP Control Protocol (RTCP) Extension for a Third-Party Loss Report", RFC 6642,

June 2012.

- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.
- [RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.
- [RFC6773] Phelan, T., Fairhurst, G., and C. Perkins, "DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal", RFC 6773, November 2012.
- [RFC6787] Burnett, D. and S. Shanmugham, "Media Resource Control Protocol Version 2 (MRCPv2)", RFC 6787, November 2012.
- [RTCP-FB] "Session Description Protocol (SDP) RTCP Feedback attributes", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-14>>.
- [SSRC-GROUP] "Session Description Protocol (SDP) "ssrc-group" semantics", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-17>>.
- [T.38] "Procedures for real-time Group 3 facsimile communication over IP networks", <<http://www.itu.int/rec/T-REC-T.38/e>>.

Author's Address

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2014

S. Nandakumar
C. Jennings
Cisco
July 12, 2013

SDP for the WebRTC
draft-nandakumar-rtcweb-sdp-02

Abstract

The Web Real-Time Communication (WebRTC) [WEBRTC] working group is charged to provide protocol support for direct interactive rich communication using audio, video and data between two peers' web browsers. With in the WebRTC framework, Session Description protocol (SDP) [RFC4566] is used for negotiating session capabilities between the peers. Such a negotiataion happens based on the SDP Offer/Answer exchange mechanism described in the RFC 3264 [RFC3264].

This document serves a introductory purpose in describing the role of SDP for the most common WebRTC use-cases.

This SDP examples provided in this document is still a work in progress, but aims to align closest to the evolving standards.

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 January 13, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
 (<http://trustee.ietf.org/license-info>) in effect on the date of
 publication of this document. Please review these documents
 carefully, as they describe your rights and restrictions with respect
 to this document. Code Components extracted from this document must
 include Simplified BSD License text as described in Section 4.e of
 the Trust Legal Provisions and are provided without warranty as
 described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. SDP and the WebRTC	3
4. Offer/Answer and the WebRTC	4
5. WebRTC Session Description Examples	5
5.1. Secure Two-Way Audio,Video and Data with RTCP Feedback . .	5
5.2. Secure Two-way Audio,Video,Data and remove data stream . .	11
5.3. Secure Two-Way Audio,Video with BUNDLE Support Unknown . .	15
5.4. Secure Two-Way Audio,Video w/BUNDLE Support Known	22
5.5. Secure Two-Way Audio,Video w/BUNDLE Unsupported	26
5.6. Successful One Way Session with 2 Video Streams	30
5.7. Sendonly Simulcast w/2 cameras and 2 encodings per camera	34
5.8. Successful SVC Video Stream	40
5.9. Successful Simulcast Video Streams with Retransmission . .	44
5.10. Successful 1-way Simulcast with 2 resolutions and RTX - One resolution rejected	48
5.11. Simulcast Video Stream with Forward Error Correction . . .	52
6. WebRTC <-> Legacy Interop Examples	56
6.1. Successful legacy Interop Fallaback with bundle-only . . .	56
7. IANA Considerations	61
8. Change Log	61
9. References	62
9.1. Normative References	62
9.2. Informative References	62
Authors' Addresses	64

1. Introduction

Javascript Session Exchange Protocol(JSEP) [JSEP] specifies a generic protocol needed to generate [RFC3264] offers and answers negotiated between the WebRTC peers for setting up, updating and tearing down a multimedia session. For this purpose, SDP is used to construct [RFC3264] offers/answers for describing (media and non-media) streams as appropriate for recipients of a session description to participate in the session.

The remainder of this document is organized as follows: Section 3 and 4 provides an overview of SDP and the Offer/Answer exchange mechanism. Section 5 and 6 provide sample SDP usages for the most common WebRTC use-cases.

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].

3. SDP and the WebRTC

The purpose of this section is to provide a general overview of SDP and its components. For a more in-depth understanding, the readers are advised to refer to [RFC4566].

The Session Description Protocol (SDP) [RFC4566] describes multimedia sessions, which can be audio, video, whiteboards, fax, modem, and other streams. It provides a general purpose, standard representation to describe various aspects of multimedia session such as media capabilities, transport addresses and related metadata in a transport agnostic manner, for the purposes of session announcement, session invitation and parameter negotiation.

As of today SDP is widely used in the context of Session Initiation Protocol, Real-time Transport Protocol and Real-time Streaming Protocol.

Below figure introduces high-level breakup of SDP into components that semantically describe a multimedia session, in our case, say, a WebRTC session [WEBRTC]. It by no means captures everything about SDP and hence, should be used for informational purposes only.

[WEBRTC] proposes JavaScript application to fully control the signaling plane of a multimedia session as described in the JSEP

specification [JSEP]. JSEP provides mechanisms to create session characterisation and media definition information to conduct the session based on SDP exchanges.

In this context,SDP serves two purposes:

- Provide grammatical structure syntatically
- Semantically convey participant's intention and capabilities.

4. Offer/Answer and the WebRTC

This section introduces SDP Offer/Answer Exchange mechanism mandated by WebRTC for negotitating session capabilities while setting up,updating and tearing down a WebRTC session. This section is intentionally brief in nature and interested readers are recommended to [RFC3264] for specific details on the protocol operation.

The Offer/Answer [RFC3264] model specifies rule for the bilateral exchange of Session Description Protocol (SDP) messages for creation of multimedia streams. It defines protocol with involved participants exchanging desired session characteristics from each others perspective modelled on SDP to negotiate the session between them.

In the most basic form,the protocol operation begins by one of the participants sending an initial SDP offer describing its intent to start a multimedia communication session. The participant receiving the offer MAY generate an SDP answer accepting the offer or it MAY reject the offer. If the session is accepted the Offer/Answer model guarantees a common view of the multimedia session between the participants.

At any time, either participant MAY generate a new SDP offer that updates the session in progress.

With in the context of WebRTC, the Offer/Answer model defines the state-machinery for WebRTC peers to negotiate session descriptions between them during the initial setup stages as well as for eventual session updates. Javascript Session Establishment Protocol specification [JSEP] for WebRTC provides the mechanism for generating [RFC3264] SDP offers and answers in order for both sides of the session to agree upon details such as list of media formats to be sent/received, bandwidth information, crypto parameters, transport parameters, for example.

The following sections provide samples of SDP message details and exchanges for the most common WebRTC usecases.

5. WebRTC Session Description Examples

A typical web based real-time multimedia communication session can be characterized as below:

- It has zero or more Audio only, Video only or Audio/Video Media streams
- MAY contain zero or more non-media data streams
- All the streams are secured with DTLS/SRTP
- ICE processing for NAT Traversal
- Sessions over IPv4-only, IPv6-only, dual-stack based clients.

As mentioned earlier [WEBRTC] proposes using SDP based Offer/Answer model to negotiate multimedia session between peers' browsers. Building on the concepts from the previous sections, the following subsections attempt to describe the usage of SDP for the most common WebRTC use-cases.

In all the use-cases, Alice and Bob are assumed to be the WebRTC peers unless mentioned otherwise. Pointers to appropriate RFCs and notes are provided, wherever necessary, against the SDP lines.

5.1. Secure Two-Way Audio, Video and Data with RTCP Feedback

This use-case allows two users to participate in a two-way communication session securely on their WebRTC enabled Web browsers.

```
title WebRTC Session - 2-Way Secure Audio, Video with RTCP Feedback
Alice->Bob: Offer(Audio:G.711,Opus,iLBC Video:H.264,VP8)
Bob->Alice: Answer(Audio:Opus,DTMF Video:H.264)
Alice->Bob: Two-way Opus Audio, H.264 Video
note right of Alice
  Session also supports RTP/RTCP Mux, RTCP feedback (nack, pli)
end note
```

More specifically, this use-case demonstrates following aspects of a WebRTC session

- SRTP with DTLS based encryption
- RTP and RTCP Muxing
- RTCP based feedback and reduced size support
- ICE processing for NAT Traversal
- Audio Codec Offered : PCMU, Opus, iLBC
- Audio Codec Answered : Opus
- Video Codecs Offered: H.264, VP8
- Video Codecs Answered: H.264
- Data Channel Support

The tables (5.1 and 5.2) below capture in detail, the initial SDP Offer and Answer messages exchanged.

The exact SDP parameters specified for Data-Channel is still under the WG discussion and is expected to be updated once the final decision is reached..

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20518 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245] - Session Level ICE parameter
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245] - Session Level ICE parameter
a=fingerprint:sha-1	[RFC5245] - Session
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:7	DTLS Fingerprint for
0:9d:1f:66:79:a8:07	SRTP
a=rtcp-rsize	[RFC5506] - Alice intends to use reduced size RTCP for this session
m=audio 54609 RTP/SAVPF 0 109 98	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:0 PCMU/8000	[RFC3551] G.711 Audio Codec
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rt p-opus] - Opus Codec 48khz, 2 channels
a=ptime:20	[draft-ietf-payload-rt p-opus] - Opus packetization of 20ms
a=rtpmap:98 iLBC/8000	[RFC3952] - Internet Low Bitrate Codec
a=fmtp:98 mode=20	[RFC3952]
a=sendrecv	[RFC3264] - Alice can send and recv audio
a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing on port 54609
b=AS:64	[RFC4566] - Audio Session B/W of 64kbps

b=RS:800	[RFC3556] - RTCP b/w allocated to active data senders
b=RR:2400	[RFC3556] - RTCP b/w allocated to receivers
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245] - Host ICE Candidate
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245] - Server Reflexive ICE Candidate for the above host candidate
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245] - Second Host Candidate
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245] - Server Reflexive Candidate for the Second Host Candidate
a=rtcp-fb:109 nack	[RFC5104] - Indicates NACK RTCP feedback support
m=video 62537 RTP/SAVPF 99 120	[RFC4566] Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=msid:ma tb	
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:99 H264/90000	[RFC3984] - H.264 Video Codec
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1	[RFC3984]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-vp8] - VP8 video codec
a=sendrecv	[RFC3264] - Alice can send and recv video
a=rtcp-mux	[RFC5761] - Alice can perform RTP/RTCP Muxing on port 62537
b=AS:256	[RFC4566] - Audio Session B/W of 256kbps
b=RS:800	[RFC3556] - RTCP b/w allocated to active data senders
b=RR:2400	[RFC3556] - RTCP b/w allocated to receivers
a=candidate:0 1 UDP 2113667327 192.168.1.4 62537 typ host	[RFC5245]

a=candidate:1 1 UDP 1694302207 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62537	[RFC5245]
a=candidate:0 2 2113667326 192.168.1.4 54721 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 54721 typ srflx raddr 192.168.1.4 rport 54721	[RFC5245]
a=rtcp-fb:99 nack pli	[RFC5104] - Indicates support for Picture loss Indication and NACK
a=rtcp-fb:99 ccm fir	[RFC5104] - Full Intra Frame Request-Codec Control Message support
a=rtcp-fb:120 nack pli	[RFC5104] - Indicates support for Picture loss Indication and NACK
a=rtcp-fb:120 ccm fir	[RFC5104] - Full Intra Frame Request-Codec Control Message support
m=application 56966 DTLS/SCTP 5000	[draft-ietf-rtcweb-dat a-channel]
c= IN IP4 24.23.204.141	[RFC4566]
a=sctpmap:5000 webrtc-Datachannel 1	[draft-ietf-mmusic-sct p-sdp] - One data stream of type chat
a=webrtc-Datachannel:5000 stream=1;label="channel 1";subprotocol="chat";	[draft-ietf-mmusic-sct p-sdp]
a=sendrecv	[RFC3264] - Alice can send and recv non-media data
a=candidate:0 1 UDP 2113667327 192.168.1.7 56966 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 24.23.204.141 56966 typ srflx raddr 192.168.1.7 rport 56966	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 51641 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 51641 typ srflx raddr 192.168.1.7 rport 51641	[RFC5245]

Table 1: 5.1 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=bob 16833 0 IN IP4 0.0.0.0	[RFC4566] - Session Origin Information
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:c300d85b	[RFC5245] - Session Level ICE username frag
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245] - Session Level ICE password
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70 :9d:1f:66:79:a8:07	[RFC5245] - Session DTLS Fingerprint for SRTP
a=rtcp-rsize	[RFC5506] - Alice intends to use reduced size RTCP for this session
m=audio 49203 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus] - Bob accept only Opus Codec
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendrecv	[RFC3264] - Bob can send and recv audio
a=rtcp-mux	[RFC5761] - Bob can perform RTP/RTCP Muxing on port 49203
b=AS:64	[RFC4566] - Audio Session b/w of 64kbps
b=RS:800	[RFC3556] - RTCP b/w allocated to active data senders
b=RR:2400	[RFC3556] - RTCP b/w allocated to receivers

a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245] - Host ICE Candidate for Opus Stream
a=ccandidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245] - Server Reflexive ICE Candidate for the above host candidate
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245] - Second Host Candidate
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245] - Server Reflexive Candidate for the Second Host Candidate
m=video 63130 RTP/SAVPF 99 a=msid:ma tb	[RFC4566] Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 98.248.92.771 a=rtpmap:99 H264/90000	[RFC4566] [RFC3984] - Bob accepts H.264 Video Codec.
a=fmtp:99 profile-level-id=4d0028;packetization-mod e=1 a=sendrecv	[RFC3984]
a=rtcp-mux	[RFC3264] - Bob can send and recv video
b=AS:256	[RFC5761] - Bob can perform RTP/RTCP Muxing on port 63130
b=RS:800	[RFC4566] - Audio Session B/W of 256kbps
b=RR:2400	[RFC3556] - RTCP b/w allocated to active data senders
a=candidate:0 1 UDP 2113667327 192.168.1.7 63130 typ host	[RFC3556] - RTCP b/w allocated to receivers
a=candidate:1 1 UDP 1694302207 98.248.92.77 63130 typ srflx raddr 192.168.1.7 rport 63130	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 56607 typ host	[RFC5245]

a=candidate:1 2 UDP 1694302206 98.248.92.77 56607 typ srflx raddr 192.168.1.7 rport 56607 a=rtcp-fb:99 nack pli	[RFC5245]
a=rtcp-fb:99 ccm fir	[RFC5104] - Indicates support for Picture loss Indication and NACK [RFC5104] - Full Intra Frame Request-Codec Control Message support
m=application 55700 DTLS/SCTP 5000	[draft-ietf-mmusic-sctp-sdp]
c= IN IP4 98.248.92.771 a=sctpmap:5000 webrtc-Datachannel 1	[RFC4566] [draft-ietf-mmusic-sctp-sdp]
a=webrtc-Datachannel:5000 stream=1;label="channel 1";subprotocol="chat"; a=sendrecv	[draft-ietf-mmusic-sctp-sdp]
a=candidate:0 1 UDP 2113667327 192.168.1.7 55700 typ host a=candidate:1 1 UDP 1694302207 98.248.92.77 55700 typ srflx raddr 192.168.1.7 rport 55700 a=candidate:0 2 UDP 2113667326 192.168.1.7 58137 typ host a=candidate:1 2 UDP 1694302206 98.248.92.77 58137 typ srflx raddr 192.168.1.7 rport 581371	[RFC3264] - Bob can send and recv non-media data [RFC5245] - Refer 4.1 SDP Offer [RFC5245] Refer 4.1 SDP Offer [RFC5245] Refer 4.1 SDP Offer [RFC5245] Refer 4.1 SDP Offer

Table 2: 5.1 SDP Answer

5.2. Secure Two-way Audio,Video,Data and remove data stream

This scenario builds upon from the usecase in the section 5.1 It extends by Alice removing data-stream once the session is in progress.

There is an ongoing discussion with in the working group to allow addition and deletion of streams using partial Offer/Answer exchanges based on m=lines. Once a final decision is reached, the following example shall be updated to reflect the same.

```

title WebRTC Session (Audio,Video,Datachannel) - Drop Datachannel
note right of Alice
    Alice & Bob are in a two-way audio,video and datachannel session.
    Alice decides to stop the datachannel stream
end note
Alice->Bob: Offer(Audio:Opus Video:VP8, Application: Drop)
Bob->Alice: Answer(Audio:Opus Video:VP8, Application:Drop)
Alice->Bob: Two-way Opus Audio and VP8 Video

```

As a precondition, A Two-Way Audio,Video and Data Session is already setup.

SDP Contents	RFC#/Notes
v=1	[RFC4566] Incremented version to indicate the update
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufraq:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70 :9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=audio 54609 RTP/SAVPF 0 109 98	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=rtpmap:98 iLBC/8000	[RFC3952] - Internet Low Bitrate Codec
a=fmtp:98 mode=20	[RFC3952]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]

a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-fb:109 nack	[RFC5104]
m=video 62537 RTP/SAVPF 99 120	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:99 H264/90000	[RFC3984]
a=fmtp:99 profile-level-id=4d0028;packetization-mod e=1	[RFC3984]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-v p8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 62537 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62537	[RFC5245]
a=candidate:0 2 2113667326 192.168.1.4 54721 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 54721 typ srflx raddr 192.168.1.4 rport 54721	[RFC5245]
a=rtcp-fb:99 nack pli	[RFC5104]
a=rtcp-fb:99 ccm fir	[RFC5104]
a=rtcp-fb:120 nack pli	[RFC5104]
a=rtcp-fb:120 ccm fir	[RFC5104]
m=application 0 DTLS/SCTP 5000	[draft-ietf-mmusic-sc tp-sdp] - Port 0 indicates dropping data stream
c= IN IP4 24.23.204.141	[RFC4566]
a=sctmap:5000 webrtc-Datachannel 1	[draft-ietf-mmusic-sc tp-sdp]
a=webrtc-Datachannel:5000 stream=1;label="channel 1";subprotocol="chat";	[draft-ietf-mmusic-sc tp-sdp]

a=sendrecv	[RFC3264]	
+-----+-----+-----+		

Table 3: 5.2 SDP Updated Offer w/DataChannel Drop

SDP Contents	RFC#/Notes
v=1	[RFC4566]
o=bob 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70 :9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=audio 49203 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-r tp-opus]
a=ptime:20	[draft-ietf-payload-r tp-opus]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=ccandidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
m=video 63130 RTP/SAVPF 99	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)

c= IN IP4 98.248.92.771	[RFC4566]
a=rtpmap:99 H264/90000	[RFC3984]
a=fmtp:99	[RFC3984]
profile-level-id=4d0028;packetization-mode=1	
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.7 63130 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 63130 typ srflx raddr 192.168.1.7 rport 63130	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 56607 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 56607 typ srflx raddr 192.168.1.7 rport 56607	[RFC5245]
a=rtcp-fb:99 nack pli	[RFC5104]
a=rtcp-fb:99 ccm fir	[RFC5104]
m=application 0 DTLS/SCTP 5000	[draft-ietf-mmusic-sctp-sdp] Bob accepts dropping the data stream
c= IN IP4 98.248.92.771	[RFC4566]
a=sctpmap:5000 webrtc-Datachannel 1	[draft-ietf-mmusic-sctp-sdp]
a=webrtc-Datachannel:5000 stream=1;label="channel 1";subprotocol="chat";	[draft-ietf-mmusic-sctp-sdp]
a=sendrecv	[RFC3264]

Table 4: 5.2 SDP Updated Answer

5.3. Secure Two-Way Audio, Video with BUNDLE Support Unknown

This use-case demonstrates a successful audio and video media streams multiplexing scenario using SDP BUNDLE negotiation framework [draft-ietf-mmusic-sdp-bundle-negotiation]

Since Alice is unsure of the Bob's support for BUNDLE framework, the SDP example below shows

- o An SDP Offer, in which the Alice assigns unique addresses to each "m=" line in the BUNDLE group, and requests the Answerer to select the Offerer's BUNDLE address.
- o An SDP Answer, in which the Bob selects the BUNDLE address for the Offerer, and assigns its own local BUNDLE address to each "m=" line in the BUNDLE group.

- o A subsequent SDP Offer from Alice, which is used to perform a BUNDLE Address Synchronization (BAS).
Once the Offer/Answer exchange completes, both Alice and Bob each end up using single RTP Session for both the Media Streams.

title WebRTC Session - 2-Way Secure Audio,Video with BUNDLE

Alice->Bob: Offer(Audio:Opus Video:VP8) with BUNDLE support and unique addresses.

Bob->Alice: Answer(Audio:Opus Video:VP8) indicating its support for BUNDLE

Alice->Bob: Updated Offer(Audio:Opus Video:VP8) for Bundle Address Synchronzation

.

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
a=group:BUNDLE foo bar	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 109	[RFC4566]
a=mid:foo	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:200	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 62537 RTP/SAVPF 120	[RFC4566]

a=mid:bar	[RFC5888] Video m=line part of the Bundle group with a unique port number
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:1756	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:22222	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 62537 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62537	[RFC5245]
a=candidate:0 2 2113667326 192.168.1.4 54721 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 54721 typ srflx raddr 192.168.1.4 rport 54721	[RFC5245]

Table 5: 5.3 SDP Offer w/BUNDLE

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=bob 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efb abd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
a=group:BUNDLE foo bar	[draft-ietf-mmusic-sdp-bun dle-negotiation] Bob supports BUNDLE semantics
m=audio 49203 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:foo	[RFC5888] Audio m=line part of the BUNDLE group
c= IN IP4 98.248.92.77	[RFC4566]
b=AS:200	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-op us]
a=ptime:20	[draft-ietf-payload-rtp-op us]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:33333	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
m=video 49203 RTP/SAVPF 120	[RFC4566]

a=mid:bar	[RFC5888] Video m=line part of the BUNDLE group with the port from audio line repeated
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
b=AS:1756	[RFC4566]
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:44444	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245] - Candidate lines identical with the audio m-line.
a=candidate:1 1 UDP 1694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]

Table 6: 5.3 SDP Answer w/BUNDLE

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
a=group:BUNDLE foo bar	[draft-ietf-mmusic-sdp-bundle-negotiation]
m=audio 54609 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:foo	[RFC5888] - Port number finalized as Bundle Address.
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:200	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 54609 RTP/SAVPF 120	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)

a=mid:bar	[RFC5888] - Same Bundle address from the audio m=line
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:1756	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:22222	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245] - Candidate lines identical with the audio m-line.
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]

Table 7: 5.3 SDP Offer for BAS

5.4. Secure Two-Way Audio,Video w/BUNDLE Support Known

This use-case is a successful audio and video stream multiplexing scenario, with Alice and Bob aware of each others support for SDP BUNDLE framework [draft-ietf-mmusic-sdp-bundle-negotiation].

title WebRTC Session - 2-Way Secure Audio,Video with BUNDLE Support Known
 Alice->Bob: Offer(Audio:Opus Video:VP8) with identical port numbers
 Bob->Alice: Answer(Audio:Opus Video:VP8) with identical port numbers
 Alice->Bob: Two-way Opus Audio, H.264 Video over a single 5-tuple

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
a=group:BUNDLE foo bar	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics.
m=audio 10000 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:foo	[RFC5888] - Audio m=line part of BUNDLE group
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:200	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 10000 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 10000 typ srflx raddr 192.168.1.4 rport 10000	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 10000 RTP/SAVPF 120	[RFC4566]

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=mid:bar	[RFC5888] - Video m=line with Bundle address same as the audio m=line
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:1000	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:22222	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 10000 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 10000 typ srflx raddr 192.168.1.4 rport 10000	[RFC5245]
a=candidate:0 2 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]

Table 8: 5.4 SDP Offer w/BUNDLE

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=bob 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efb abd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
a=group:BUNDLE foo bar	[draft-ietf-mmusic-sdp-bun dle-negotiation] - Bob supports BUNDLE semantics
m=audio 20000 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:foo	[RFC5888] - Audio m=line part of the BUNDLE group
b=AS:200	[RFC4566]
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-op us]
a=ptime:20	[draft-ietf-payload-rtp-op us]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:33333	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.7 20000 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 20000 typ srflx raddr 192.168.1.7 rport 20000	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]
m=video 20000 RTP/SAVPF 120	[RFC4566]

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=mid:bar	[RFC5888] - Video m=line with Bundle address same as the audio m=line
b=AS:1000	[RFC4566]
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:44444	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.7 20000 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 98.248.92.77 20000 typ srflx raddr 192.168.1.7 rport 20000	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 98.248.92.77 60065 typ srflx raddr 192.168.1.7 rport 60065	[RFC5245]

Table 9: 5.4 SDP Answer w/BUNDLE

5.5. Secure Two-Way Audio,Video w/BUNDLE Unsupported

This use-case illustrates SDP Offer/Answer exchange when the far-end (Bob) either doesn't support media bundling or doesn't want to group m=lines over a single 5-tuple.

On successful Offer/Answer exchange, Alice and Bob each end up using separate RTP sessions for audio and video media streams respectively.

title WebRTC Session - 2-Way Secure Audio,Video with BUNDLE Unsupported

Alice->Bob: Offer(Audio:Opus Video:VP8) with BUNDLE support, unique port numbers

Bob->Alice: Answer(Audio:Opus Video:VP8) with no BUNDLE support, unique port numbers

Alice->Bob: Two-way Opus Audio, H.264 Video over 2 different RTP sessions.

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20518 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
a=group:BUNDLE foo bar	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 55232 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=mid:foo	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:200	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 55232 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 55232 typ srflx raddr 192.168.1.4 rport 55232	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 54332 RTP/SAVPF 120	[RFC4566]

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=mid:bar	[RFC5888] Video m=line part of the BUNDLE group with a unique port number
c= IN IP4 24.23.204.141	[RFC4566]
b=AS:1000	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-vp8]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:22222	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54332 typ host	[RFC5245]
a=candidate:1 1 UDP 1694302207 24.23.204.141 54332 typ srflx raddr 192.168.1.4 rport 54332	[RFC5245]
a=candidate:0 2 2113667326 192.168.1.4 54721 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 54721 typ srflx raddr 192.168.1.4 rport 54721	[RFC5245]

Table 10: 5.5 SDP Offer w/BUNDLE

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=bob 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1	[RFC5245]
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	
a=rtcp-rsize	[RFC5506]
m=audio 53214 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
b=AS:200	[RFC4566]
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-r tp-opus]
a=ptime:20	[draft-ietf-payload-r tp-opus]
a=sendrecv	[RFC3264]
a=candidate:0 1 UDP 2113667327	[RFC5245]
192.168.1.7 53214 typ host	
a=candidate:1 1 UDP 1694302207	[RFC5245]
98.248.92.77 53214 typ srflx raddr	
192.168.1.7 rport 53214	
a=candidate:0 2 UDP 2113667326	[RFC5245]
192.168.1.7 60065 typ host	
a=candidate:1 2 UDP 1694302206	[RFC5245]
98.248.92.77 60065 typ srflx raddr	
192.168.1.7 rport 60065	
m=video 58679 RTP/SAVPF 120	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
b=AS:2000	[RFC4566]
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-v p8]
a=sendrecv	[RFC3264]

a=candidate:0 1 UDP 2113667327 192.168.1.7 58679 typ host	[RFC5245]	
a=candidate:1 1 UDP 1694302207 98.248.92.77 58679 typ srflx raddr 192.168.1.7 rport 58679	[RFC5245]	
a=candidate:0 2 UDP 2113667326 192.168.1.7 56607 typ host	[RFC5245]	
a=candidate:1 2 UDP 1694302206 98.248.92.77 56607 typ srflx raddr 192.168.1.7 rport 56607	[RFC5245]	

Table 11: 5.5 SDP Answer without BUNDLE

5.6. Successful One Way Session with 2 Video Streams

In this scenario Alice and Bob engage in one-way multimedia session with Bob receiving two video streams, one corresponding to Alice's video and other corresponding to her presentation slides.

```

title 1 Way Audio & Video w/2 Video Streams
note right of Alice
Alice offers 2 sendonly video streams
one for her video feed and other for her presentation slides.
end note
Alice->Bob: Offer(Audio:Opus, Video1,2: VP8)
note right of Bob
Bob accepts Alice's offer
end note
Bob->Alice: Answer(Audio:Opus, Video1,2: VP8)
Alice->Bob: One-way Opus Audio, VP8 Video
note right of Alice
Bob can hear Alice and see her video feed as well
as her presentation slides.
end note

```


SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70 :9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=audio 54609 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-r tp-opus]
a=ptime:20	[draft-ietf-payload-r tp-opus]
a=sendonly	[RFC3264] - Send only audio stream
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5104]
m=video 62537 RTP/SAVPF 120	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-v p8]
a=content:speaker	[RFC4796] - Stream 1 for Alice's video
a=sendonly	[RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 62537 typ host	[RFC5245]

a=candidate:0 2 UDP 2113667326 192.168.1.4 54721 typ host m=video 62539 RTP/SAVPF 120 a=msid:mb ta	[RFC5245]
	[RFC4566]
	Identifies RTCMediaStream ID (mb) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-v p8]
a=content:slides	[RFC4796] - Stream 2 for Alice's slides
a=sendonly	[RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 62539 typ host	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 54723 typ host	[RFC5245]

Table 12: 5.6 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=bob 16833 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70 :9d:1f:66:79:a8:07	[RFC5245]
a=rtcp-rsize	[RFC5506]
m=audio 49203 RTP/SAVPF 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 98.248.92.77	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-r tp-opus]
a=ptime:20	[draft-ietf-payload-r tp-opus]
a=recvonly	[RFC3264] - Receive only audio stream
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 60065 typ host	[RFC5245]
m=video 63130 RTP/SAVPF 120	[RFC4566]
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 98.248.92.771	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-v p8]
a=content:speaker	[RFC4796] - Stream 1 for Alice's Video
a=recvonly	[RFC3264] - Receive Only Video Stream 1
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.7 63130 typ host	[RFC5245]

a=candidate:0 2 UDP 2113667326 192.168.1.7 56607 typ host m=video 63133 RTP/SAVPF 120 a=msid:mb ta	[RFC5245]
	[RFC4566]
	Identifies RTCMediaStream ID (mb) and RTCMediaStreamTrack ID (ta)
c= IN IP4 98.248.92.771	[RFC4566]
a=rtpmap:120 VP8/90000	[draft-ietf-payload-v p8]
a=content:slides	[RFC4796] - Stream 2 for Alice's Slides
a=recvonly	[RFC3264] - Receive Only Video Stream 2
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.7 63133 typ host	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.7 56609 typ host	[RFC5245]

Table 13: 5.6 SDP Answer

5.7. Sendonly Simulcast w/2 cameras and 2 encodings per camera

This SDP below shows Offer/Answer exchange with audio and two video streams each of which can send two different resolutions.

One video stream supports VP8, while the other supports H.264.

bundle-only framework is used along with BUNDLE grouping framework to enable multiplexing of all the 5 streams (audio stream + 4 video streams) over a single RTP Session.

```

title 1 Way Successful Simulcast
note right of Alice
Alice offers 2 sendonly video streams with 2 simulcast encodings per stream
end note
Alice->Bob: Offer(Audio:Opus,Video1:VP8,Video2:H.264) with bundle-only for video
note left of Bob
Bob accepts Alice's offer and 2 encodings per stream
end note
Alice->Bob: One-Way 1 Opus, 2 H.264 and 2 VP8 video streams.
```

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=group:BUNDLE m0 m1 m2	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 0 RTP/SAVPF 98 100	bundle-only video line with port number set to zero

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]Camera-1,Encoding -1 Resolution
a=fmtp:98 max-fr=30	[RFC4566]
a=imageattr:100 [x=640,y=480]	[RFC6236] Camera-1,Encoding-2 Resolution
a=fmtp:100 max-fr=15	[RFC4566]
a=ssrc-group:SIMULCAST 12345 45678	[RFC5576]
a=ssrc:12345	[RFC5576]
a=ssrc:45678	[RFC5576]
a=ssrc:12345 cname:axzo1278npDlAzM73	[RFC5576] [draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-1 SSRC with Session CNAME
a=ssrc:45678 cname:axzo1278npDlAzM73	[RFC5576] [draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-2 SSRC with Session CNAME
a=sendonly	[RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available
m=video 0 RTP/SAVPF 98 100	bundle-only video line with port number set to zero
a=msid:ma tc	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tc)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m2	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 H264/90000	[RFC3984]
a=rtpmap:100 H264/90000	[RFC3984]
a=fmtp:98	[RFC3984]Camera-2,Encoding -1 Resolution
profile-level-id=4d0028;packetization-mode=1;max-fr=30	

a=fmtp:100 profile-level-id=4d0028;packetization-mode=1;max-fr=15	[RFC3984]Camera-1,Encoding-2 Resolution
a=ssrc:67890	[RFC5576]
a=ssrc:56789	[RFC5576]
a=ssrc-group:SIMULCAST 67890 56789	[RFC5576]
a=ssrc:67890 cname:axzo1278npDlAzM73	[RFC5576]
a=ssrc:56789 cname:axzo1278npDlAzM73	[draft-rescorla-avtcore-6222bis] Camera-1,Encoding-1 SSRC with Session CNAME [RFC5576]
a=sendonly	[draft-rescorla-avtcore-6222bis] Camera-1,Encoding-2 SSRC with Session CNAME [RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 14: 5.8 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efb abd9a2	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=group:BUNDLE m0 m1 m2	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number

a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:22222	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 54609 RTP/SAVPF 98 100	BUNDLE accepted with port repeated from the audio port
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]Camera-1,Encoding-1 Resolution
a=fmtp:98 max-fr=30	[RFC4566]
a=imageattr:100 [x=640,y=480]	[RFC6236]Camera-1,Encoding-2 Resolution
a=fmtp:100 max-fr=15	[RFC4566]
a=ssrc-group:SIMULCAST 54321 87654	[RFC5576]
a=ssrc:54321	[RFC5576]
a=ssrc:87654	[RFC5576]
a=ssrc:54321 cname:axzo1278npDlAzM73	[RFC5576]
	[draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-1 SSRC with Session CNAME

a=ssrc:87654 cname:axzo1278npDlAzM73	[RFC5576] [draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-2 SSRC with Session CNAME
a=sendonly	[RFC3264] - Send only video stream
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-mux	[RFC5576]
a=bundle-only	Add reference to unified plan when available
m=video 54609 RTP/SAVPF 98 100	BUNDLE accepted with port repeated from the audio port
c= IN IP4 24.23.204.141	[RFC4566]
a=msid:ma tc	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tc)
a=mid:m2	[RFC5888] Video m=line part of BUNDLE group
a=rtpmap:98 H264/90000	[RFC3984]
a=rtpmap:10 H264/90000	[RFC3984]
a=fmtp:98 profile-level-id=4d0028;packetizatio n-mode=1;max-fr=30	[RFC3984]Camera-2,Encoding -1 Resolution
a=fmtp:100 profile-level-id=4d0028;packetizatio n-mode=1;max-fr=15	[RFC3984]Camera-1,Encoding -2 Resolution
a=ssrc:90876	[RFC5576]
a=ssrc:89754	[RFC5576]
a=ssrc-group:SIMULCAST 90876 89754	[RFC5576]
a=ssrc:90876 cname:axzo1278npDlAzM73	[RFC5576] [draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-1 SSRC with Session CNAME
a=ssrc:89754 cname:axzo1278npDlAzM73	[RFC5576] [draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-2 SSRC with Session CNAME

a=sendonly	[RFC3264] - Send only
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	video stream [RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
a=rtcp-mux	[RFC5576]
a=bundle-only	Add reference to unified plan when available

Table 15: 5.8 SDP Answer

5.8. Successful SVC Video Stream

This section shows an SDP Offer/Answer for a session with an audio and a single video stream. The video stream is layered coding at 3 different resolutions based on [RFC5583]. The video m=line shows 3 streams with last stream (payload 100) dependent on streams with payload 96 and 97 for decoding.

```

title 2 way SVC Video
note right of Alice
Alice offers 3 sendonly video stream with 3 layers of SVC
end note
Alice->Bob: Offer(Audio:Opus Video: H.264 SVC) bundle-only
note left of Bob
Bob accepts Alice's Offered Codec operation points
end note
Bob->Alice: Answer(Video: H.264) bundle-only
Alice->Bob: One-Way H.264 Video with codec points as indicated by Bob.
```

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c 7:70:9d:1f:66:79:a8:07	[RFC5245]
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 0 RTP/SAVPF 96 97 100	bundle-only video line with port number set to zero

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tc)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m1	[RFC5888] Audio m=line part of BUNDLE group
a=msid:ma tb	Add appropriate reference when available
b=AS:2500	[RFC4566]
a=rtpmap:96 H264/90000	[RFC3984]
a=fmtp:96	[RFC3984]H.264 Layer 1
profile-level-id=4d0028;packetization-mode=1;max-fr=30;max-fs=8040	
a=rtpmap:97 H264/90000	[RFC3984]
a=fmtp:97	[RFC3984] H.264 Layer 2
profile-level-id=4d0028;packetization-mode=1;max-fr=15;max-fs=1200	
a=rtpmap:100 H264-SVC/90000	[RFC3984]
a=fmtp:100	[RFC3984]
profile-level-id=4d0028;packetization-mode=1;max-fr=30;max-fs=8040	
a=depend:100 lay m1:96,97;	[RFC5583]Layer 3 dependent on layers 1 and 2
a=sendonly	[RFC3264] - Send only video stream
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 16: 5.9 SDP Offer with SVC

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1	[RFC5245]
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.142	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.5 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.142 54609 typ srflx raddr 192.168.1.5 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.5 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.142 64678 typ srflx raddr 192.168.1.5 rport 64678	[RFC5245]
m=video 54609 RTP/SAVPF 96 100	BUNDLE accepted Bundle address same as audio m=line.

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.142	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
b=AS:2500	[RFC4566]
a=rtpmap:96 H264/90000	[RFC3984]
a=fmtp:96	[RFC3984]H.264 Layer 1
profile-level-id=4d0028;packetization-mode=1;max-fr=30;max-fs=8040	
a=rtpmap:100 H264-SVC/90000	[RFC3984]
a=fmtp:100	[RFC3984]
profile-level-id=4d0028;packetization-mode=1;max-fr=30;max-fs=8040	
a=depend:100 lay m1:96;	[RFC5583] Bob chooses 2 Codec Operation points
a=candidate:0 1 UDP 2113667327 192.168.1.5 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.142 54609 typ srflx raddr 192.168.1.5 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.5 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.142 64678 typ srflx raddr 192.168.1.5 rport 64678	[RFC5245]
a=recvonly	[RFC3264] - Receive only video stream
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 17: 5.9 SDP Answer with SVC

5.9. Successful Simulcast Video Streams with Retransmission

This section shows an SDP Offer/Answer exchange for a simulcast scenario with 2 two resolutions and has [RFC4588] style retransmission flows.

```

title Simulcast Streams with Retransmission
note right of Alice
Alice offers single audio and simulcasted video stream
end note
Alice->Bob: Offer(Audio:Opus Video:VP8 with 2 resolutions and RTX Stream) bundle-
only
note right of Bob
Bob accepts all the streams offered by Alice
end note
Bob->Alice: Answer(Audio:Opus Video:VP8 with 2 resolutions and RTX Stream) bundle
-only
note right of Alice
Successful 2 way simulcast session with associated retransmission streams
end note

```

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bun- dle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-op- us]
a=ptime:20	[draft-ietf-payload-rtp-op- us]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]

a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 0 RTP/SAVPF 98 100 101 103	bundle-only video line with port number set to zero
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:ml	[RFC5888] Audio m=line part of BUNDLE group
b=AS:1756	[RFC4566]
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:101 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:103 VP8/90000	[draft-ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000	[RFC4588]
a=fmtp:103 apt=100;rtx-time=3000	[RFC4588]
a=ssrc-group:SIMULCAST 12345 78990	Simulcast group
a=ssrc-group:FID 12345 34567	[RFC5888]
a=ssrc-group:FID 78990 90887	[RFC5888]
a=ssrc:12345	[RFC5576]
a=ssrc:78990	[RFC5576]
a=ssrc:34567	[RFC5576]
a=ssrc:90887	[RFC5576]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 18: 5.10 SDP Offer w/Simulcast, RTX

SDP Contents	RFC#/Notes
v=0	[RFC4566]

o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1	[RFC5245]
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.142	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:33333	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.5 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.142 54609 typ srflx raddr 192.168.1.5 rport 54609	
a=candidate:0 2 UDP 2113667326 192.168.1.5 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.142 64678 typ srflx raddr 192.168.1.5 rport 64678	[RFC5245]
m=video 54609 RTP/SAVPF 98 100 101 103	BUNDLE accepted with Bundle address identical to audio m-line
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)

c= IN IP4 24.23.204.142	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
b=AS:1756	[RFC4566]
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:101 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:103 VP8/90000	[draft-ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000	[RFC4588]
a=fmtp:103 apt=100;rtx-time=3000	[RFC4588]
a=candidate:0 1 UDP 2113667327 192.168.1.5 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.142 54609 typ srflx raddr 192.168.1.5 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.5 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.142 64678 typ srflx raddr 192.168.1.5 rport 64678	[RFC5245]
a=ssrc-group:SIMULCAST 54321 77656	Simulcast group
a=ssrc-group:FID 54321 88776	[RFC5888]
a=ssrc-group:FID 77656 12908	[RFC5888]
a=ssrc:54321	[RFC5576]
a=ssrc:77656	[RFC5576]
a=ssrc:88776	[RFC5576]
a=ssrc:12908	[RFC5576]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 19: 5.10 SDP Answer w/Simulcast, RTX

5.10. Successful 1-way Simulcast with 2 resolutions and RTX - One resolution rejected

This section shows an SDP Offer/Answer exchange for a simulcast scenario with 2 two resolutions and has [RFC4588] style re-transmission flows.

It also showcases when Bob rejects one of the Simulcast Video Stream which results in the rejection of the associated repair stream implicitly

```

title Simulcast Streams with Retransmission
note right of Alice
Alice offers single audio and simulcasted video stream
end note
Alice->Bob: Offer(Audio:Opus Video:VP8 with 2 resolutions and RTX Streams) bundle
-only
note right of Bob
Bob accepts one simulcast,rtx and rejects the other
end note
Bob->Alice: Answer(Audio:Opus Video:VP8 with 1 resolution and the RTX Stream) bun
dle-only
note right of Alice
Successful 1 way session with 1 Video Stream and the associated RTX Stream
end note

```

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bun dle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-op us]
a=ptime:20	[draft-ietf-payload-rtp-op us]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]

a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 0 RTP/SAVPF 98 100 101 103	bundle-only video line with port number set to zero
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:ml	[RFC5888] Audio m=line part of BUNDLE group
b=AS:1756	[RFC4566]
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:101 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:103 VP8/90000	[draft-ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000	[RFC4588]
a=fmtp:103 apt=100;rtx-time=3000	[RFC4588]
a=ssrc-group:SIMULCAST 12345 78990	Simulcast group
a=ssrc-group:FID 12345 34567	[RFC5888]
a=ssrc-group:FID 78990 90887	[RFC5888]
a=ssrc:12345	[RFC5576]
a=ssrc:78990	[RFC5576]
a=ssrc:34567	[RFC5576]
a=ssrc:90887	[RFC5576]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 20: 5.11 SDP Offer w/Simulcast, RTX

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1	[RFC5245]
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.142	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=recvonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:33333	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.5 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.142 54609 typ srflx raddr 192.168.1.5 rport 54609	
a=candidate:0 2 UDP 2113667326 192.168.1.5 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.142 64678 typ srflx raddr 192.168.1.5 rport 64678	[RFC5245]
m=video 54609 RTP/SAVPF 98 101	BUNDLE accepted with Bundle address identical to audio m-line

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.142	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
b=AS:1756	[RFC4566]
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:101 VP8/90000	[draft-ietf-payload-vp8]
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:101 apt=98;rtx-time=3000	[RFC4588]
a=candidate:0 1 UDP 2113667327 192.168.1.5 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.142 54609 typ srflx raddr 192.168.1.5 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.5 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.142 64678 typ srflx raddr 192.168.1.5 rport 64678	[RFC5245]
a=ssrc-group:FID 54321 88776	[RFC5888]
a=ssrc:54321	[RFC5576]
a=ssrc:88776	[RFC5576]
a=recvonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 21: 5.11 SDP Answer no Simulcast

5.11. Simulcast Video Stream with Forward Error Correction

This section shows an SDP Offer/Answer exchange for Simulcast video stream at two resolutions and has [RFC5956] style FEC flows.

On completion of Offer/Answer exchange, one audio stream, 2 simulcast video streams and 2 associated FEC streams are sent over a single 5-Tuple as part of bundle-only and BUNDLE framework.

title Simulcast Streams with Forward Error Correction
 note right of Alice
 Alice offers single audio and simulcasted video stream
 end note
 Alice->Bob: Offer(Audio:Opus Video:VP8 with 2 resolutions with FEC Streams) bundle-only
 note right of Bob
 Bob accepts simulcast stream as well as FEC streams
 end note
 Bob->Alice: Answer(Audio:Opus Video:VP8 with 2 resolutions with FEC Streams) bundle-only
 note right of Alice
 Successful Session with 4 video streams (Simulcast + FEC) and 1 Audio Stream

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufraq:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9: c7:70:9d:1f:66:79:a8:07	[RFC5245]
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:11111	[RFC5576]

a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 0 RTP/SAVPF 98 100 101 103	bundle-only video line with port number set to zero
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
b=AS:2500	[RFC4566]
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:101	[RFC5956]
ld-interleaved-parityfec/90000	
a=rtpmap:103	[RFC5956]
ld-interleaved-parityfec/90000	
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 L=5; D=10; repair-window=200000	[RFC5956]
a=fmtp:103 L=5; D=10; repair-window=200000	[RFC5956]
a=ssrc-group:SIMULCAST 12345 78990	Simulcast group
a=ssrc-group:FEC-FR 12345 34567	[RFC5888]
a=ssrc-group:FEC-FR 78990 90887	[RFC5888]
a=ssrc:12345	[RFC5576]
a=ssrc:78990	[RFC5576]
a=ssrc:34567	[RFC5576]
a=ssrc:90887	[RFC5576]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 22: 5.12 SDP Offer

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1	[RFC5245]
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	
a=group:BUNDLE m0 m1	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=ssrc:33333	[RFC5576]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 54609 RTP/SAVPF 98 100 101 103	BUNDLE accepted with Bundle Address identical to audio m=line.

a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
b=AS:2500	[RFC4566]
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:101	[RFC5956]
ld-interleaved-parityfec/90000	
a=rtpmap:103	[RFC5956]
ld-interleaved-parityfec/90000	
a=fmtp:98 max-fr=30;max-fs=8040	[RFC4566]
a=fmtp:100 max-fr=15;max-fs=1200	[RFC4566]
a=fmtp:101 L=5; D=10;	[RFC5956]
repair-window=200000	
a=fmtp:103 L=5; D=10;	[RFC5956]
repair-window=200000	
a=ssrc-group:SIMULCAST 54321 77656	Simulcast group
a=ssrc-group:FEC-FR 54321 88776	[RFC5888]
a=ssrc-group:FEC-FR 77656 12908	[RFC5888]
a=ssrc:54321	[RFC5576]
a=ssrc:77656	[RFC5576]
a=ssrc:88776	[RFC5576]
a=ssrc:12908	[RFC5576]
a=sendrecv	[RFC3264]
a=rtcp-mux	[RFC5761]
a=bundle-only	Add reference to unified plan when available

Table 23: 5.12 SDP Offer

6. WebRTC <-> Legacy Interop Examples

In this section, we attempt to provide session descriptions showcasing inter-operability between a WebRTC end-point and a Legacy VOIP end-point. The ideas included in here are not fully baked into the standards yet.

6.1. Successful legacy Interop Fallaback with bundle-only

In the scenario described below, Alice is a multi-stream capable WebRTC endpoint while Bob is a legacy VOIP end-point. The SDP Offer/Answer exchange demonstrates successful session setup with fallback

to audio only stream negotiated via bundle-only framework between the end-points. Specifically,

- o Offer from Alice describes 2 cameras via 2 video m=lines with both marked as bundle-only.
- o Since Bob doesnot recognize either BUNDLE mechanism or bundle-only attribute, he accepts only the audio stream from Alice.

title Successful 2-Way WebRTC <-> VOIP Interop

note right of Alice

Alice is a multistream capable WebRTC end-point & Bob is behind a legacy VOIP system

end note

Alice->Bob: Offer(Audio:Opus Video: 2 VP8, 2 H2.64 Streams) with bundle-only offer

note right of Alice

Alice marks both the video streams as bundle-only

end note

Bob->Alice: Answer(Audio:Opus)

note right of Bob

Bob accepts only Audio stream since he doesn't recognize bundle-only streams

end note

Alice->Bob: Two-way Opus Audio

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:074c6550	[RFC5245]
a=ice-pwd:a28a397a4c3f31747dlee3474af08a068	[RFC5245]
a=fingerprint:sha-1	[RFC5245]
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	
a=group:BUNDLE m0 m1 m2	[draft-ietf-mmusic-sdp-bundle-negotiation] Alice supports grouping of m=lines under BUNDLE semantics
m=audio 54609 RTP/SAVPF 0 109	[RFC4566]
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m0	[RFC5888] Audio m=line part of BUNDLE group with a unique port number

a=msid:ma ta	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (ta)
a=rtpmap:0 PCMU/8000	[RFC3551]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=rtcp-mux	[RFC5761]
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	[RFC5245]
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	[RFC5245]
a=candidate:0 2 UDP 2113667326 192.168.1.4 64678 typ host	[RFC5245]
a=candidate:1 2 UDP 1694302206 24.23.204.141 64678 typ srflx raddr 192.168.1.4 rport 64678	[RFC5245]
m=video 0 RTP/SAVPF 98 100	bundle-only video line with port number set to zero
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m1	[RFC5888] Video m=line part of BUNDLE group
a=msid:ma tb	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tb)
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]Camera-1,Encoding-1 Resolution
a=fmtp:98 max-fr=30	[RFC4566]
a=imageattr:100 [x=640,y=480]	[RFC6236]Camera-1,Encoding-2 Resolution
a=fmtp:100 max-fr=15	[RFC4566]
a=ssrc-group:SIMULCAST 12345 45678	[RFC5576]
a=ssrc:12345	[RFC5576]
a=ssrc:45678	[RFC5576]
a=ssrc:12345 cname:axzo1278npDlAzM73	[RFC5576]
	[draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-1 SSRC with Session CNAME

a=ssrc:45678 cname:axzo1278npDlAzM73	[RFC5576]
a=bundle-only	[draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-2 SSRC with Session CNAME Add reference to unified plan when available
m=video 0 RTP/SAVPF 101 103	bundle-only video line with port number set to zero
c= IN IP4 24.23.204.141	[RFC4566]
a=mid:m2	[RFC5888] Video m=line part of BUNDLE group
a=msid:ma tc	Identifies RTCMediaStream ID (ma) and RTCMediaStreamTrack ID (tc)
a=rtpmap:101 H264/90000	[RFC3984]
a=rtpmap:103 H264/90000	[RFC3984]
a=fmtp:101 profile-level-id=4d0028;packetizatio n-mode=1;max-fr=30	[RFC3984]Camera-2,Encoding -1 Resolution
a=fmtp:100 profile-level-id=4d0028;packetizatio n-mode=1;max-fr=15	[RFC3984]Camera-1,Encoding -2 Resolution
a=ssrc:67890	[RFC5576]
a=ssrc:56789	[RFC5576]
a=ssrc-group:SIMULCAST 67890 56789	[RFC5576]
a=ssrc:67890 cname:axzo1278npDlAzM73	[RFC5576]
a=ssrc:56789 cname:axzo1278npDlAzM73	[draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-1 SSRC with Session CNAME
a=ssrc:56789 cname:axzo1278npDlAzM73	[RFC5576]
a=ssrc:56789 cname:axzo1278npDlAzM73	[draft-rescorla-avtcore-62 22bis] Camera-1,Encoding-2 SSRC with Session CNAME
a=ssrc:56789 cname:axzo1278npDlAzM73	Add reference to unified plan when available

Table 24: 6.1 SDP Simulcast bundle-only

SDP Contents	RFC#/Notes
v=0	[RFC4566]
o=alice 20519 0 IN IP4 0.0.0.0	[RFC4566]
s=	[RFC4566]
t=0 0	[RFC4566]
a=ice-ufrag:ufrag:c300d85b	[RFC5245]
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	[RFC5245]
a=fingerprint:sha-1	[RFC5245]
99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	
m=audio 54609 RTP/SAVPF 109	[RFC4566]
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:109 opus/48000/2	[draft-ietf-payload-rtp-opus]
a=ptime:20	[draft-ietf-payload-rtp-opus]
a=sendonly	[RFC3264]
a=candidate:0 1 UDP 2113667327	[RFC5245]
192.168.1.4 54609 typ host	
a=candidate:1 1 UDP 694302207	[RFC5245]
24.23.204.141 54609 typ srflx raddr	
192.168.1.4 rport 54609	
a=candidate:0 2 UDP 2113667326	[RFC5245]
192.168.1.4 64678 typ host	
a=candidate:1 2 UDP 1694302206	[RFC5245]
24.23.204.141 64678 typ srflx raddr	
192.168.1.4 rport 64678	
m=video 0 RTP/SAVPF 98 100	Bob doesn't recognize bundle-only and hence rejects the video stream
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:98 VP8/90000	[draft-ietf-payload-vp8]
a=rtpmap:100 VP8/90000	[draft-ietf-payload-vp8]
a=imageattr:98 [x=1280,y=720]	[RFC6236]Camera-1,Encoding-1 Resolution
a=fmtp:98 max-fr=30	[RFC4566]
a=imageattr:100 [x=640,y=480]	[RFC6236]
	Camera-1,Encoding-2 Resolution
a=fmtp:100 max-fr=15	[RFC4566]

m=video 0 RTP/SAVPF 98 100	Bob doesn't recognize bundle-only and hence rejects the video stream
c= IN IP4 24.23.204.141	[RFC4566]
a=rtpmap:101 H264/90000	[RFC3984]
a=rtpmap:103 H264/90000	[RFC3984]
a=fmtp:101	[RFC3984]Camera-2,Encoding-1 Resolution
profile-level-id=4d0028;packetization-mode=1;max-fr=30	
a=fmtp:100	[RFC3984]Camera-1,Encoding-2 Resolution
profile-level-id=4d0028;packetization-mode=1;max-fr=15	

Table 25: 6.2 SDP Answer

7. IANA Considerations

This document requires no actions from IANA.

8. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-nandakumar-rtcweb-sdp-01

- o Updated references to OPUS RTP Payload Specification.
- o Updated BUNDLE examples based on the latest draft-ietf-mmusic-sdp-bundle-negotiation.
- o Added examples for multiple audio and video flows based on Unified Plan.
- o Added new examples for RTX and FEC streams
- o Updated Simulcast and SVC examples

Changes from draft-nandakumar-rtcweb-sdp-00

- o Fixed editorial comments on the mailing list.
- o Updated Data-channel SDP information based on draft-ietf-mmusic-sctp-sdp.
- o Updated BUNDLE examples based on draft-ietf-mmusic-sdp-bundle-negotiation.
- o Added examples for few more BUNDLE variants
- o Added new examples for Simulcast and SVC

9. References

9.1. Normative References

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

9.2. Informative References

- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, July 2006.
- [WEBRTC] W3C, "WebRTC 1.0: Real-time Communication Between Browsers",
<<http://dev.w3.org/2011/webrtc/editor/webrtc.html>> .
- [JSEP] Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-01 (work in progress), December 2012.
- [RFC5506] Johansson, I., "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 3551, July 2003.
- [RFC3952] Duric, A. and S. Andersen, "Real-time Transport Protocol (RTP) Payload Format for internet Low Bit Rate Codec (iLBC) Speech", RFC 3952, December 2004.
- [RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth",

RFC 3556, July 2003.

- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [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.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "RTP Payload Format for H.264 Video", RFC 5888, June 2010.
- [RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.
- [draft-ietf-payload-rtp-opus]
Spittka, J., Vos, K., and JM. Valin, "RTP Payload Format for Opus Speech and Audio Codec",
draft-ietf-payload-rtp-opus-00 (work in progress),
July 2012.
- [draft-ietf-payload-vp8]
Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video",
draft-ietf-payload-vp8-05 (work in progress), August 2012.
- [RFC3984] Wenger, S., Hannuksela, M., Stockhammer, T., Westerlund, M., and D. Singer, "RTP Payload Format for H.264 Video", RFC 3984, February 2005.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [draft-ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "RTCWeb Datagram Connection", draft-ietf-rtcweb-data-channel-01 (work in progress), September 2012.

[draft-ietf-mmusic-sctp-sdp]

Loreto, S. and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-03 (work in progress), September 2012.

[draft-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-04 (work in progress), February 2013.

[draft-lennox-mmusic-sdp-source-selection]

Lennox, J. and H. Schulzrinne, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-lennox-mmusic-sdp-source-selection-05 (work in progress), October 2012.

[draft-rescorla-avtcore-6222bis]

Rescorla, E. and A. Begen, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", draft-rescorla-avtcore-6222bis-00 (work in progress), October 2012.

Authors' Addresses

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: January 12, 2014

T. Reddy
P. Patil
D. Wing
Cisco
July 11, 2013

Happy Eyeballs Extension for ICE
draft-reddy-mmusic-ice-happy-eyeballs-01

Abstract

This document specifies requirements for algorithms that make ICE connectivity checks more responsive by reducing delays in dual-stack host ICE connectivity checks when there is a path failure for the address family preferred by the application or by the operating system. As IPv6 is usually preferred, the procedures in this document helps avoid user-noticeable delays when the IPv6 path is broken or excessively slow.

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 January 12, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Notational Conventions	2
3. Candidates Priority	2
4. Algorithm overview	3
4.1. Processing the Results	5
5. Indicating Happy-Eyeballs	6
6. IANA Considerations	6
7. Security Considerations	7
8. Acknowledgements	7
9. References	7
9.1. Normative References	7
9.2. Informative References	8
Authors' Addresses	8

1. Introduction

In situations where there are many IPv6 addresses, ICE [RFC5245] will prefer IPv6 candidates [RFC6724] and will attempt connectivity checks on all the IPv6 candidates before trying an IPv4 candidate. If the IPv6 path is broken, this fallback to IPv4 can consume a lot of time, harming user satisfaction of dual-stack devices.

This document describes an algorithm that makes ICE connectivity checks more responsive to failures of an address family by reordering the candidate pairs such that IPv6 and IPv4 candidates get a fair chance during connectivity checks. This document specifies requirements for any such algorithm, with the goals that the ICE agent need not be inordinately harmed with a simple reordering of the candidate pairs.

2. Notational Conventions

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

This note uses terminology defined in [RFC5245].

3. Candidates Priority

A prioritization formula is used by ICE [RFC5245] so that most preferred address pairs are tested first, and if a sufficiently good

pair is discovered, the tests can be stopped. With IPv6, addresses obtained from local network interfaces, called host candidates, are recommended as high-priority ones to be tested first since if they work, they provide usually the best path between the two hosts. The ICE specification recommends to use the rules defined in [RFC6724] as part of the prioritization formula for IPv6 host candidates and [I-D.keranen-mmusic-ice-address-selection] updates the ICE rules on how IPv6 host candidates are selected.

For dual-stack hosts the preference for IPv6 host candidates is higher than IPv4 host candidates based on precedence value of IP addresses described in [RFC6724]. IPv6 server reflexive candidates have higher precedence than IPv4 server reflexive candidate since NPTv6 is stateless and transport-agnostic.

```
(highest)  IPv6 Host Candidate
           IPv4 Host Candidate
           IPv6 Server Reflexive Candidate
           IPv4 Server Reflexive Candidate
           IPv6 Relayed Transport Candidate
(lowest)   IPv4 Relayed Transport Candidate
```

Figure 1: Candidate Preferences in decreasing order

By using the technique described in Section 4, if there are both IPv6 and IPv4 addresses in the check list, and the first 'N' candidates are of the same IP address family, then the highest-priority candidate pair of the other address family is promoted to position N in the check list thus making ICE connectivity checks more responsive to failures of an address family.

Note: The algorithm works even if the administrator changes the policy table to prefer IPv4 addresses over IPv6 addresses as defined in [RFC6724]

4. Algorithm overview

The Happy Eyeballs Extension for ICE algorithm proposes the following:

1. Indicate support for ICE Happy Eyeballs in the SDP offer if the end-point is dual-stack. The end-point will also include the position 'N' at which promotion is to occur.

2. After SDP offer/answer exchange if both end points support ICE Happy Extension for ICE algorithm the following steps are performed by the ICE agents after computing the candidate pair priority, ordering and pruning the pairs (section 5.7.2, 5.7.3 of [RFC5245])
 - a. If the first 'N' candidate pairs in the check list are of the same IP address family, then the highest-priority candidate pair of the other address family is promoted to position 'N' in the list.
 - b. Step b is repeated for candidate pairs that are next in the check list. This is continued until all candidate pairs of the preferred address family are exhausted.

The result of these steps is that after every consecutive 'N' candidate pairs of the preferred family, a candidate pair of the other family is inserted.

The following figure illustrates the result of the algorithm on candidate pairs:

Before Happy Eyeballs Extension for ICE algorithm :

```
-----
(highest)  IPv6 Host Candidate-1 pair
           IPv6 Host Candidate-2 pair
           IPv6 Host Candidate-3 pair
           IPv6 Host Candidate-4 pair
           IPv6 Host Candidate-5 pair
           IPv6 Host Candidate-6 pair
           IPv6 Host Candidate-7 pair
           IPv4 Host Candidate pair
           IPv6 Server Reflexive Candidate pair
           IPv4 Server Reflexive Candidate pair
           IPv6 Relayed Transport Candidate pair
(lowest)   IPv4 Relayed Transport Candidate pair
```

After Happy Eyeballs Extension for ICE algorithm :

```
-----
(highest)  IPv6 Host Candidate-1 pair
           IPv6 Host Candidate-2 pair
           IPv6 Host Candidate-3 pair
           IPv4 Host Candidate pair          ---> Promoted pair
           IPv6 Host Candidate-4 pair
           IPv6 Host Candidate-5 pair
           IPv6 Host Candidate-6 pair
           IPv4 Server Reflexive Candidate pair ---> Promoted pair
```

```

IPv6 Host Candidate-7 pair
IPv6 Server Reflexive Candidate pair
IPv6 Relayed Transport Candidate pair
(lowest) IPv4 Relayed Transport Candidate pair

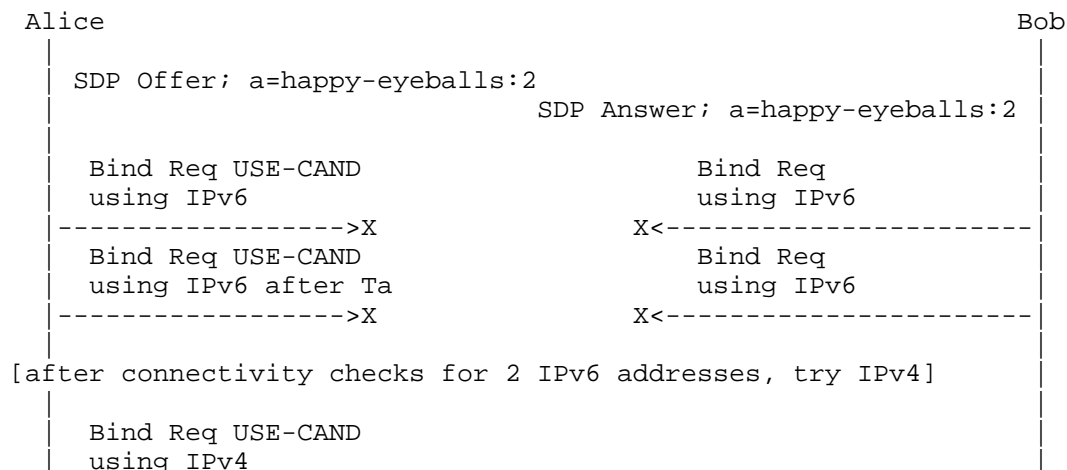
```

4.1. Processing the Results

If ICE connectivity checks using an IPv4 candidate is successful then ICE Agent will performs as usual "Discovering Peer Reflexive Candidates" (Section 7.1.3.2.1 of [RFC5245]), "Constructing a Valid Pair" (Section 7.1.3.2.2 of [RFC5245]), "Updating Pair States" (Section 7.1.3.2.3 of [RFC5245]), "Updating the Nominated Flag" (Section 7.1.3.2.4 of [RFC5245]).

If ICE connectivity checks using an IPv4 candidate is successful for each component of the media stream and connectivity checks using IPv6 candidates is not yet successful, the ICE endpoint will declare victory, conclude ICE for the media stream and start sending media using IPv4. However, it is also possible that ICE endpoint continues to perform ICE connectivity checks with IPv6 candidate pairs and if checks using higher-priority IPv6 candidate pair is successful then media stream can be moved to the IPv6 candidate pair. Continuing to perform connectivity checks can be useful for subsequent connections, to optimize which connectivity checks are tried first. Such optimization is out of scope of this document.

The following diagram shows the behaviour during the connectivity check when Alice calls Bob and Agent Alice is the controlling agent and uses the aggressive nomination algorithm. "USE-CAND" implies the presence of the USE-CANDIDATE attribute.



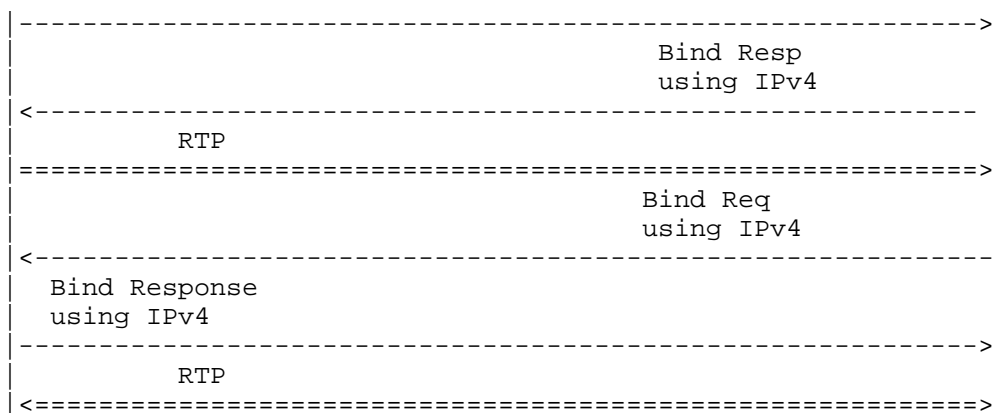


Figure 2: Happy Eyeballs Extension for ICE

5. Indicating Happy-Eyeballs

To indicate that Happy Eyeballs Extension for ICE algorithm defined in this document is used, the ICE offerer MUST include ice-options attribute with "happy-eyeballs:N" option identifier in the Session Description Protocol (SDP) [RFC4566] ICE offer, where N indicates the position at which promotion is to occur. If the ICE offer does not include this option tag, the answerer SHOULD NOT utilize the updated ICE algorithm defined in this document. If the offer included the option tag and the answerer supports this specification, the answerer SHOULD add the same option tag to the response and use the Happy Eyeballs Extension for ICE algorithm. If the ICE answer does not contain the option tag, the offerer SHOULD NOT use the updated ICE algorithm. Recommended value for 'N' is 3. As IPv6 becomes more prevalent, the value of 'N' can be increased as desired.

6. IANA Considerations

IANA is requested to register "happy-eyeballs" option identifier under the "ICE Options" [RFC6336] registry.

The required registration information is as follows:

Option identifier: happy-eyeballs

Contact: Tirumaleswar Reddy, tiredddy@cisco.com

Change control: IETF

Description: Existence of this option identifier indicates that Happy Eyeballs Extension for ICE algorithm is used.

Reference: RFCXXXX

[RFC editor: replace XXXX with the RFC number of this document]

7. Security Considerations

STUN connectivity check using MAC computed during key exchanged in the signaling channel provides message integrity and data origin authentication as described in section 2.5 of [RFC5245] apply to this use.

8. Acknowledgements

The authors would like to thank Bernard Aboba for his inputs.

9. References

9.1. Normative References

- [I-D.keranen-mmusic-ice-address-selection]
Keraenen, A. and J. Arkko, "Update on Candidate Address Selection for Interactive Connectivity Establishment (ICE)", draft-keranen-mmusic-ice-address-selection-01 (work in progress), July 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [RFC6336] Westerlund, M. and C. Perkins, "IANA Registry for Interactive Connectivity Establishment (ICE) Options", RFC 6336, July 2011.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

9.2. Informative References

- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

Authors' Addresses

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tiredddy@cisco.com

Prashanth Patil
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marthalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: praspati@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 16, 2014

A. B. Roach
Mozilla
J. Uberti
Google
M. Thomson
Microsoft
July 15, 2013

A Unified Plan for Using SDP with Large Numbers of Media Flows
draft-roach-mmusic-unified-plan-00

Abstract

A recurrent theme in emerging real-time communications use cases, such as RTCWEB, 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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Design Goals	4
1.1.1. Support for a large number of arbitrary sources	4
1.1.2. Support for fine-grained receiver control of sources	5
1.1.3. Glareless addition and removal of sources	5
1.1.4. Interworking with other devices	5
1.1.5. Avoidance of excessive port allocation	6
1.1.6. Simple binding of MediaStreamTrack to SDP	6
1.1.7. Support for RTX, FEC, simulcast, layered coding	6
1.2. Terminology	6
1.3. Syntax Conventions	7
2. Solution Overview	7
3. Detailed Description	8
3.1. Bundle-Only M-Lines	8
3.2. Correlation	12
3.2.1. Correlating RTP Sources with m-lines	12
3.2.1.1. RTP Header Extension Correlation	13
3.2.1.2. Payload Type Correlation	14
3.2.2. Correlating Media Stream Tracks with m-lines	16
3.2.3. Correlating Media Stream Tracks with RTP Sources	16
3.3. Handling of Simulcast, Forward Error Correction, and Retransmission Streams	16
3.4. Glare Minimization	18
3.4.1. Adding a Stream	19
3.4.2. Changing a Stream	19
3.4.3. Removing a Stream	20
3.5. Negotiation of Stream Ordinality	20
3.6. Compatibility with Legacy uses	22
4. Examples	23
4.1. Simple example with one audio and one video	23
4.2. Multiple Videos	26
4.3. Many Videos	28
4.4. Multiple Videos with Simulcast	30
4.5. Video with Simulcast and RTX	31
4.6. Video with Simulcast and FEC	32
4.7. Video with Layered Coding	33
5. Security Considerations	35
6. IANA Considerations	35
7. Acknowledgements	35
8. References	35

8.1. Normative References	35
8.2. Informative References	36
Authors' Addresses	37

1. Introduction

A recurrent theme in new RTC technologies has been the need to cleanly handle very large numbers of media flows. For instance, a videoconferencing 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.

This document focuses on the WebRTC use cases, and uses its terminology to discuss key concepts. The approach described herein, however, is not intended to be WebRTC specific, and should be generalized to other SDP-using applications.

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=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 52595
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 96 97
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 56036
a=rtpmap:96 H264/90000
```

```
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 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 96 97
a=msid:ma tc
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 21909
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 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.

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 (inventing as few new mechanisms as possible) while simply reducing the consumption of network resources.

1.1. Design Goals

The mechanism described in this document is meant to address the following goals:

1.1.1. 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:

- o A single large main speaker with thumbnails for other participants
- o Multiple medium-sized main speakers, with or without thumbnails
- o 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 we attempt to satisfy is support for sending and receiving up to hundreds of simultaneous, heterogeneous sources.

1.1.2. 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 each (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.

1.1.3. 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.

1.1.4. Interworking with other devices

When interacting with devices that do not apply all of the techniques described in this document, it must be possible to degrade gracefully to a usable basic experience. At a minimum, this basic experience should support setting up one audio stream and more than one video stream with existing videoconferencing equipment designed to establish a small number of simultaneous audio and video flows. For the remainder of this document, we will call these devices "legacy devices," although it should be understood that statements about legacy devices apply equally to future devices that elect not to use the techniques described in this document.

1.1.5. Avoidance of excessive 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.

1.1.6. 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.

1.1.7. 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 heterogeneous receivers. It needs to be possible to support these techniques, allow the recipient to optionally use or not use them on a source-by-source basis; and for simulcast/layered scenarios, to control which simulcast streams or layers are received.

1.2. 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.

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.

1.3. 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:

- o 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.
- o 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.
- o 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).
- o 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.

2. Solution Overview

At a high level, the solution described in this document can be summarized as follows:

1. Each media stream track is represented by its own unique m-line. This is a strict one-to-one mapping; a single media stream track cannot be spread across several m-lines, nor may a single m-line represent multiple media stream tracks. Note that this requires a modification to the way simulcast is currently defined by the individual draft [I-D.westerlund-avtcore-rtp-simulcast]. This does not preclude "application level" simulcasting; i.e., the creation of multiple media stream tracks from a single source.
2. Each m-line is marked with an a=ssrc attribute to correlate it with its RTP packets. Absent any other signaled extension, multiple SSRCs in a single m-line are interpreted as alternate

sources for the same media stream track: although senders can switch between the SSRCs as frequently as desired, only one should be sent at any given time.

3. Each m-line contains an MSID value to correlate it with a Media Stream ID and the Media Stream Track ID.
4. To minimize port allocation during a call, we rely on the BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] mechanism.
5. To reduce port allocation during call set-up, applications can mark less-critical media stream tracks in such a way that they will not require any port allocation, with the resulting property that such streams only work in the presence of the BUNDLE mechanism.
6. To address glare, we define a procedure via which partial offer/answer exchanges may take place. These exchanges operate on a single m-line at a time, rather than an entire SDP body. These operations are defined in a way that can completely avoid glare for stream additions and removals, and which reduces the chance of glare for changes to active streams. This approach requires all m-lines to contain an a=mid attribute.
7. All sources in a single bundle are required to contain identical attributes except for those that apply directly to a media stream track (such as label, msid, and resolution). See those attributes marked "IDENTICAL" in [I-D.nandakumar-mmusic-sdp-mux-attributes] for details.
8. RTP and RTCP streams are demultiplexed strictly based on their SSRC. However, to handle legacy cases and signaling/media races, correlation of streams to m-sections can use other mechanisms, as described in Section 3.2.

3. Detailed Description

3.1. Bundle-Only M-Lines

Even with the use of 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.

OPEN ISSUE: While it's probably pretty clear that this behavior will be controlled, in WebRTC, via a constraint, the "default" behavior -- that is, whether a line is "bundle-only" when there is no constraint present -- needs to be settled. This is a balancing

act between maximizing interoperability with legacy equipment by default or minimizing port use during call setup by default.

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

- o Sets the port in the m-line to 0. This indicates to old endpoints that the m-line is not to be negotiated.
- o 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=group:BUNDLE S1 S2 S3
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=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 20970
a=mid:1
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=ssrc:53280
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 96 97
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 1714
a=mid:2
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 VP8/90000
a=sendrecv
a=rtcp-mux
a=ssrc:49152
a=bundle-only
```

```
m=video 0 RTP/SAVPF 96 97
a=msid:ma tc
a=extmap:1 urn:ietf:params:rtp-hdext:stream-correlator 57067
a=mid:3
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 VP8/90000
a=sendrecv
a=rtcp-mux
a=ssrc:32768
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 96 97
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 VP8/90000
a=sendrecv
```

```
m=video 0 RTP/SAVPF 96 97
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 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=group:BUNDLE B1 B2 B3
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=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 24860
a=mid:1
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=ssrc:35987
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host

m=video 55400 RTP/SAVPF 96 97
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 49811
a=mid:B2
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 VP8/90000
a=sendrecv
a=rtcp-mux
a=ssrc:9587
a=bundle-only

m=video 55400 RTP/SAVPF 96 97
a=msid:ma tc
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 9307
a=mid:3
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:97 VP8/90000
a=sendrecv
```

```
a=rtcp-mux
a=ssrc:21389
a=bundle-only
```

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

3.2. Correlation

The system under consideration has three constructs the need to be mutually correlated for proper functioning: m-lines, media stream tracks, and RTP sources. These correlations are described in the following sections.

3.2.1. Correlating RTP Sources with m-lines

Sending several media streams over a single transport 5-tuple can pose challenges in the form of stream identification and correlation. This proposal maintains the use of SSRC as the single demultiplexing point for multiple streams sent between a transport 5-tuple.

Nominally, this correlation is performed by including a=ssrc attributes in the SDP. Under ideal circumstances, the use of a=ssrc in the SDP exchanged between endpoints is sufficient to correlate a demultiplexed stream to its m-line. However, at least three unrelated situations can arise that make correlation using an alternate mechanism advantageous.

During call establishment, circumstances may arise under which an endpoint can send an offer for a new stream, and begin receiving that media stream prior to receiving the SDP that correlates its SSRC to the m-line. For such cases, the endpoint will not know how to handle the media, and will most probably be forced to discard it. This can lead to media stream "clipping," which has a strongly negative impact on user experience. For audio streams, the "hello" of the answering party can be lost; for video streams, the initial I-frame can be lost, leading to corrupted or missing video until another I-frame is sent.

In the rare circumstance that a SSRC change for an existing media source 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. Since any media sent with the new SSRC cannot be rendered until the new offer/answer exchange takes place, the clipping concern mentioned above exists here as well.

A different problem can arise when interoperating with legacy equipment. A number of circumstances can lead to the inability of a legacy endpoint to include SSRC information in its SDP. For example, in a system that decomposes signaling and media into different network devices, the protocol used to communicate between the boxes frequently will not include SSRC information, making it impossible to include in the SDP. If these devices choose to implement bundling, correlation of media streams to m-lines requires an alternate correlator.

These cases (and possibly other similar situations) can be ameliorated by using information in the media stream itself as a correlator to the SDP offer. If a packet arrives with an SSRC that is not yet associated with an m-line, we would ideally have some means of correlating it prior to the arrival of the answer.

The authors reiterate and emphasize that this technique is used solely for the purposes of correlation of an RTP stream to an SDP m-line after that stream has already been demultiplexed. Demultiplexing of multiple streams on a single transport address continues to be based on SSRC values.

3.2.1.1. RTP Header Extension Correlation

The preferred mechanism for such correlation is a new RTP header extension [RFC5285] that can be used near the beginning of an RTP stream to correlate RTP packets for which SSRC mapping information is not available. We propose that WebRTC implementations **MUST** implement this mechanism. We expect and that all other users of the BUNDLE extension **SHOULD** make use of it.

Although additional specification for this mechanism would be required for interoperability, the thumbnail sketch of such correlation is described below.

An implementation making use of this mechanism for local correlation includes an a=extmap attribute in the m-lines for which it wishes to use the mechanism. This attribute includes a mapping from the RTP header ID to the URL, as well as a 16-bit identifier (expressed as an integer) used for correlation; one such m-line would look like this:

```
m=audio 55400 RTP/SAVPF 0 96
a=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrex:stream-correlator 7582 // NEW
a=mid:1
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
```



```
a=sendrecv
a=rtcp-mux
a=ssrc:35987
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
```

The remote endpoint, if it supports this extension, MUST include an RTP header extension in several (on the order of 3 to 10) of the initial RTP packets in the stream. The value of this header extension will contain the correlator from the extmap line (in the above example, 7582).

3.2.1.2. Payload Type Correlation

To support implementations that cannot implement the RTP header extension described in Section 3.2.1.1 but which wish to use the BUNDLE mechanism, we allow an alternate (but less-preferred) means of correlation using payload type. This approach takes advantage of the fact that the offer contains payload types chosen by its creator, which will be present in any RTP received from the remote party. If these payload types are unique, then they can be used to reliably correlate incoming RTP streams to their m= lines.

Because of its inherent limitations, it is advisable to use other correlation techniques than PT multiplexing if at all possible. In order to accomplish this, we propose, for WebRTC, that use of this technique be controlled by an additional constraint passed to createOffer by the Web application.

If this constraint is set, the browser MUST behave as described in this section. If the constraint is not set, the browser MUST use identical PTs for the same codec values within each m-line bundle.

When such a constraint is present, implementations attempt to entirely exhaust the dynamic payload type numbering space before re-using a payload type within the scope of a local transport address. If such a constraint is present and the payload type space would ordinarily be exhausted within the scope of a local transport address, the implementation MAY (at its discretion) take any of the following actions:

1. Bind to multiple local transport addresses (using different BUNDLE groups) for the purpose of keeping the {payload type, transport address} combination unique.
2. Signal a failure to the application.

OPEN ISSUE: The above text specifically calls out "dynamic payload type numbering space," which consists of payload types 96 through 127. This is the most conservative range of payload types possible, with the greatest chance of exhaustion in normal use. In practice, it may make more sense to use a different range. The canonical description of payload type allocation strategies for RTP/AVP and its related profiles is given in section 3 of [RFC3551]. Roughly summarized: all values from 0 to 127 can be dynamically bound to codecs; codes from 96 to 127 should be preferred, followed by previously unassigned values, followed by statically assigned values. This is, however, modified by [RFC5761], which effectively eliminates payload types 64 through 95. Given these constraints, reasonable proposals (in order of most conservative to most aggressive) would include:

1. The dynamic range (96-127), for 32 usable payload types. This is meant to accommodate the most naive implementation possible, which is only capable of dynamically binding payload types in the dynamic range. Although not supported by current specifications, such limitations are suspected to exist in some modern RTP libraries.
2. The dynamic range (96-127), followed by the contiguous unassigned range (35-63), for 61 usable payload types. This approach is intended to accommodate those implementations that do not support dynamic binding for payload types for which an "audio/video" type is registered in the IANA registry.
3. The dynamic range (96-127) followed by all unassigned payload types (20-24, 27, 29, 30, and 35-63), for 69 usable payload types. This approach is intended to accommodate those implementations that are incapable of re-binding statically assigned payload types, while making use of all other available values.
4. The dynamic range (96-127) followed by all unassigned payload types (20-24, 27, 29, 30, and 35-63), followed by the statically assigned payload types (0-19, 25, 26, 28, and 31-34) for 96 usable payload types. This approach is most consistent with current IETF specifications, but is expected to cause interoperability issues with existing implementations (including libraries currently in use in early WebRTC implementations).

Note that the presence or absence of the aforementioned flag does not affect how incoming streams are correlated: if the RTP header extension for correlation is present, it is used in preference to the payload type. Conversely, if the flag is absent, and the RTP

contains no such header, then the payload type may be used for correlation inasmuch as a media line can be unambiguously identified. Of course, if the SSRC information has been made available in SDP prior to a need for stream correlation, then it can also be used for this purpose.

3.2.2. Correlating Media Stream Tracks with m-lines

Media Stream Tracks IDs are correlated with M-Lines directly by including an MSID in each m-line. The MSID also provides the Media Stream ID. (Note the format of the MSID used here is slightly different than what was proposed in the current MSID draft as that draft assumed multiple tracks in a single m-line and this proposal moves to a solution where there is a one to one relation between the Track and MSID. This work assumes the MSID draft will be updated to match the syntax used user which simply provides the value of the MediaStream ID and MediaStreamTrack ID on an "a=msid" line.)

3.2.3. Correlating Media Stream Tracks with RTP Sources

Media Stream Tracks are correlated with RTP sources transitively through the RTP-Source \Leftrightarrow M-Line \Leftrightarrow Media-Stream-Track relationship. Since the Media-Stream-Track \Leftrightarrow M-Line binding is established in the SDP offer, and the M-Line \Leftrightarrow RTP-Source binding can be handled as described in Section 3.2.1, none of the previously identified issues arise.

3.3. Handling of Simulcast, Forward Error Correction, and Retransmission Streams

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.

Forward Error Correction (FEC) and Retransmission (RTX) streams are techniques that can provide stream robustness in the face of packet loss. These approaches frequently make use of different payload types and different SSRC values than the stream to which they apply.

In cases where a media source needs to correspond to more than one RTP flow, e.g. RTX, FEC, or simulcast, the a=ssrc-group [RFC5576] concept is used to create a grouping of SSRCs for a single media stream track. 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 (modulo the behavior described in Section 3.2.1, if applicable). 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].

Note that both RTC and FEC also include SDP expressions that use different m= lines for the correction streams (cf. [RFC4588], section 8.7 and [RFC5956], section 4.2). These formats intend for correlation of streams to be based on transport addresses, which is inapplicable for bundled media streams. Our specific proposal is: (1) bundling implementations will never generate such a format; and (2) bundling implementations MAY choose to accept SDP in such a format or MAY simply reject the repair streams and proceed as if the indicated repair format is not supported.

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 [I-D.westerlund-avtcore-rtp-simulcast], as outlined in Section 2, bullet 1). In the example below, the SDP advertises a simulcast of a camera source at two different resolutions, as well as a screen-share source that supports RTX; a=ssrc-group is used to correlate the different SSRCS as part of a single media source.

Note that a characteristic of this approach is that it does not allow for independently setting attributes for simulcast, FEC, and RTX streams aside from those in fmlp. In particular, attributes such as ptime and framerate are shared between the streams that are grouped together for a simulcast group.

```
m=video 62537 RTP/SAVPF 96          // main video
a=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 15955
a=mid:1
a=rtpmap:96 VP8/90000
a=sendrecv
a=rtcp-mux
a=ssrc:29154 imageattr:96 [x=1280,y=720]
a=ssrc:47182 imageattr:96 [x=640,y=360]
a=ssrc-group:SIMULCAST 29154 47182
```

```
m=video 0 RTP/SAVPF 96 97          // slide video
```

```
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 26267
a=mid:2
a=rtpmap:96 VP8/90000
a=rtpmap:97 rtx/90000
a=sendrecv
a=rtcp-mux
a=fmtp:97 apt=96;rtx-time=3000
a=ssrc:45982
a=ssrc:9827
a=ssrc-group:FID 45982 9827          // FID provides SSRC correlation
a=bundle-only
```

Providing explicit resolutions on a per-SSRC basis for 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.

3.4. Glare Minimization

To allow for guaranteed glareless addition and removal of streams, and to provide for a reduced chance of glare in stream attribute changes, we propose a technique that allows for m-lines to be changed independently of each other.

The proposal for doing so is performed using "partial offers" and "partial answers." Using this technique has two key prerequisites: (1) all offer/answer exchanges in the session have contained "a=mid" attributes [RFC5888] for each m-line, and (2) both sides are known to support the partial offer/answer technique (either because they are part of a single domain of control, or because use of this technique has been explicitly signaled).

The use of a partial SDP body will be explicitly signaled, e.g., using a different MIME type for SIP, or using a different "type" for the WebRTC API.

The authors recognize that further formal definition would be required to describe this technique. These are left as future study for the appropriate venues, such as the W3C WebRTC WG and the SIPCORE WG. As a thumbnail sketch: For WebRTC, we envision that we would add a new constraint to `createOffer`, requesting that a partial offer be generated (if possible). The resulting `RTCSessionDescription` would contain only the m-lines that have changed since the most recent offer/answer exchange, and would have a type of "partialOffer." When `createAnswer` is called after receipt of a `partialOffer`, it would

create a partialAnswer, containing only the m-lines referenced in the partial offer, that can be provided to the remote party.

3.4.1. Adding a Stream

To add a stream glarelessly, a party creates a "partial offer" consisting of an m-line and all of its attributes. This m-line contains an mid that has not yet been used in the session. To reduce the chance of collision to effectively zero, this mid MUST contain at least 32 characters chosen randomly from full set of 79 characters allowed in a token. It then sends this partial offer to the remote party and awaits a partial answer.

Upon receipt of a partial offer, an implementation examines the mid in it. If the mid does not match any existing mid in the session, then it represents a new media stream. Assuming the recipient does not have an outstanding, unanswered partial offer that also adds a stream, this new m-line is simply appended to the end of the existing session description, the SDP version is incremented by one, and a partial answer is created. This partial answer consists of an m-line and its attributes, and has an mid matching the one from the partial offer.

If the recipient of a partial offer that contains a new mid has also sent a partial offer adding a new stream to the session, then ambiguity can arise regarding the canonical ordering of m-lines within the session. In this situation, both partial offer/answer exchanges are allowed to complete independently (as no fundamental data glare has occurred). However, the order in which they are appended to the session description is synchronized by performing a lexical comparison between each m-lines mid attribute: the m-line with the lexically smaller mid attribute is appended first, while the other m-line is appended after it.

3.4.2. Changing a Stream

Partial offers may also be generated for modification of an existing stream. In this case, the mid in the partial offer will match an existing mid in the session description.

Upon receipt of a partial offer, an implementation examines the mid in it. If the mid matches any existing mid in the session, then it represents a modification to that m-line. Assuming the recipient does not have an outstanding, unanswered partial offer that also modifies that exact same stream, this m-line is treated as an independent renegotiation of that stream (only). The SDP version is incremented by one, and a partial answer is created. This partial answer consists of an m-line and its attributes, and has an mid matching the one from the partial offer.

If the recipient of a partial offer that contains an existing mid has also sent a partial offer to change that exact same stream, and neither the received nor the sent partial offer contains an "a=inactive" attribute, then a legitimate glare condition has arisen. Normal glare recovery procedures -- e.g., using a tie-breaker token or a back-off timer -- must be engaged to resolve the conflict.

3.4.3. Removing a Stream

To remove a stream in a way that eliminates the chance of glare, an implementation generates a new partial offer, with an mid matching the m-line it wants to remove. This partial offer contains an a=inactive attribute, indicating that the stream is being deactivated.

If the recipient of a partial offer that contains an existing mid has also sent a partial offer to change that exact same stream, and either one of the received or the sent partial offer contains an "a=inactive" attribute, then the stream is deactivated. At this point, both partial offers are discarded, the corresponding m-line in the session is modified by changing any a=sendonly, a=recvonly, or a=sendrecv attribute to a=inactive (or, if no such attributes are present, an a=inactive attribute is added), and a partial answer is generated representing this single change.

3.5. Negotiation of Stream Ordinality

Within advanced applications, circumstances can easily arise in which the party creating the offer does not know ahead of time the number of streams the remote party will desire. For example, in a meet-me videoconference application that sends a separate stream for each participant, a client creating an offer to send to the conference focus does not necessarily know how many video streams to indicate in its SDP. Although this can be potentially be solved in an application-specific way (e.g., by always offering the maximum number of streams known to be supported by the application), this is not always desirable or even possible.

To address this situation, a three-way handshake can be employed.

	Calling Party	Called Party
Calling party creates offer with audio and video. Since it does not know how many streams, it "guesses" one of each.	--- Offer (1 video, 1 audio) -->	
[Call starts now]	<-- Answer (1 video, 1 audio) -- <-- Offer (8 video, 1 audio) ---	Called party desires eight video streams. So it creates an answer for the "one of each" offer and an offer for the total number of streams it wants.
Calling party answers for all eight video streams.	-- Answer (8 video, 1 audio) -->	

The first leg of this handshake consists of an offer sent by the calling party. This offer contains at least one m-line for each type of media the offerer wishes to use in the session. The authors draw special attention to the clause "at least" in the preceding sentence: offerers can use external knowledge, hinting, or simple guesses to offer additional m-lines.

Upon receipt of such an offer, the called party examines the number of streams of each media type being requested. If the number of streams is equal to or greater than the number of total streams that the called party desires at this time, it simply forms an answer to complete the offer/answer exchange [RFC3264], and the call is set up.

On the other hand, if the called party determines that more streams are necessary than are indicated in the initial offer, it responds by first creating an answer with the same number of streams as were present in the initial offer. It additionally creates a new answer that contains the number of streams it desires. This answer/offer pair is sent to the calling party, in a single message if supported by the signaling protocol (as will frequently be the case for WebRTC), or in two consecutive messages in a way that guarantees in-order delivery.

When the calling party receives this answer, it establishes the session, and all of the streams that were negotiated in this first offer/answer exchange. So, within a single signaling round trip, the initial set of streams (consisting of those the calling party included in its initial offer) are established.

When the calling party receives the subsequent offer, it comprises the beginning of a completely new RFC 3264 offer/answer exchange [RFC3264]. The calling party creates an answer that fully describes all of the streams in the session, and sends it to the called party. Consequently, within 1.5 round trips, the entire call is set up and all associated streams can be sent and received.

Of particular note is the fact that this model does not deviate from normal RFC3264 offer/answer handling, even when three-way handshaking is necessary.

3.6. Compatibility with Legacy uses

Due to the fact that this approach re-uses existing SDP constructs for indicating parameters in a media section, it remains compatible with legacy clients. Of particular note is the handling of "bundle-only" media sections, described in Section 3.1. Offers generated by an RTCWEB client and sent to a legacy 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 legacy clients (by not making them bundle-only), and which ones are not. Offers generated by legacy 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.

4. 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.

4.1. Simple example with one audio and one video

The following SDP shows an offer that offers one audio stream and one video stream with both a STUN and TURN address. It also shows unique payload across the audio and video m=lines for the Answerer that does not support BUNDLE semantics.

```
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:a28a397a4c3f31747dlee3474af08a068
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=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdext:stream-correlator 33424
a=mid:m1
a=ssrc:53280
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=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 35969
a=mid:m2
a=ssrc:49843
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 an 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=msid:ma ta
```

```
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=msid:ma tb
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 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
a=group:BUNDLE m1 m2

m=audio 60600 RTP/SAVPF 109
a=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 39829
```

```
a=mid:m1
a=ssrc:35856
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=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 45163
a=mid:m2
a=ssrc:2638
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
```

4.2. 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=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 47434
a=mid:m1
```

```
a=ssrc:32385
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 96 98
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 22705
a=mid:m2
a=ssrc:43985
a=rtpmap:96 H264/90000
a=fmtp:96 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 96 98
a=msid:ma tc
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 64870
a=mid:m3
a=ssrc:54269
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 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
```

4.3. Many Videos

This section adds three video streams and one audio. The video streams are sent in such a way that they are only accepted if the far side supports bundle using the "bundle only" approach described in Section 3.1. 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=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 6614
a=mid:m0
a=ssrc:12359
a=rtpmap:0 PCMU/8000
a=rtpmap:96 opus/48000
a=ptime:20
a=sendrecv
a=rtcp-mux
a=ssrc:12359 cname:45:5f:fe:cb:81:e9
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 96 98
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 24147
a=mid:m1
a=ssrc:26989
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:26989 cname:45:5f:fe:cb:81:e9
```

```
m=video 0 RTP/SAVPF 96 98
a=msid:ma tc
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 33989
a=mid:m2
a=ssrc:32986
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:32986 cname:45:5f:fe:cb:81:e9
```

```
m=video 0 RTP/SAVPF 96 98
a=msid:ma td
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 61408
a=mid:m3
a=ssrc:46986
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1
a=rtpmap:98 VP8/90000
a=sendrecv
a=rtcp-mux
a=bundle-only
a=ssrc:46986 cname:45:5f:fe:cb:81:e9
```


4.4. 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 3.3. One video stream supports VP8, while the other supports H.264. All the video is bundle-only. Note that the use of different codec-specific parameters causes two different payload types to be used.

```
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

m=audio 56600 RTP/SAVPF 0 96
a=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 31727
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 96 100
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 41664
b=AS:1756
a=mid:m1
a=rtpmap:96 VP8/90000
a=ssrc-group:SIMULCAST 58949 28506
a=ssrc:58949 imageattr:96 [x=1280,y=720]
a=ssrc:28506 imageattr:96 [x=640,y=480]
```

```
a=sendrecv
a=rtcp-mux
a=bundle-only

m=video 0 RTP/SAVPF 96 100
a=msid:ma tc
a=extmap:1 urn:ietf:params:rtp-hdext:stream-correlator 14460
b=AS:1756
a=mid:m2
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d0028;packetization-mode=1;max-fr=30
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=4d0028;packetization-mode=1;max-fr=15
a=ssrc-group:SIMULCAST 18875 54986
a=ssrc:18875
a=ssrc:54986
a=sendrecv
a=rtcp-mux
a=bundle-only
```

4.5. 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

m=audio 56600 RTP/SAVPF 0 96
a=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdext:stream-correlator 42123
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 96 101
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 60725
b=AS:2500
a=mid:m1
a=rtpmap:96 VP8/90000
a=rtpmap:101 rtx/90000
a=fmtp:101 apt=96;rtx-time=3000
a=ssrc-group:SIMULCAST 78909 43567
a=ssrc-group:FID 78909 56789
a=ssrc-group:FID 43567 13098
a=ssrc:78909
a=ssrc:43567
a=ssrc:13098
a=ssrc:56789
a=sendrecv
a=rtcp-mux
a=bundle-only
```

4.6. Video with Simulcast and FEC

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 [RFC5956] style FEC 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
```

```
m=audio 56600 RTP/SAVPF 0 96
a=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 42123
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 96 101
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 60725
b=AS:2500
a=mid:m1
a=rtpmap:96 VP8/90000
a=rtpmap:101 ld-interleaved-parityfec/90000
a=fmtp:96 max-fr=30;max-fs=8040
a=fmtp:101 L=5; D=10; repair-window=200000
a=ssrc-group:SIMULCAST 56780 34511
a=ssrc-group:FEC-FR 56780 48675
a=ssrc-group:FEC-FR 34511 21567
a=ssrc:56780
a=ssrc:34511
a=ssrc:21567
a=ssrc:48675
a=sendrecv
a=rtcp-mux
a=bundle-only
```

4.7. Video with Layered Coding

This section shows an SDP offer that has an audio and a single video stream. The video stream that is layered coding at 3 different resolutions based on [RFC5583]. The video m=lines shows 3 streams with last stream (payload 100) dependent on streams with payload 96 and 97 for decoding.

```
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

m=audio 56600 RTP/SAVPF 0 96
a=msid:ma ta
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 42123
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 96 97 100
a=msid:ma tb
a=extmap:1 urn:ietf:params:rtp-hdrext:stream-correlator 60725
b=AS:2500
a=mid:m1
a=rtpmap:96 H264/90000
a=fmtp:96 max-fr=30;max-fs=8040
a=rtpmap:97 H264/90000
a=fmtp:97 max-fr=15;max-fs=1200
a=rtpmap:100 H264-SVC/90000
a=fmtp:100 max-fr=30;max-fs=8040
```

```
a=depend:100 lay m1:96,97;  
a=ssrc:48970  
a=ssrc:90898  
a=ssrc:66997  
a=sendrecv  
a=rtcp-mux  
a=bundle-only
```

5. Security Considerations

TBD

6. IANA Considerations

TBD

7. Acknowledgements

Thanks to Cullen Jennings and Suhas Nandakumar for their assistance in generating the SDP examples in this document.

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

8. References

8.1. 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.
- [I-D.westerlund-avtcore-rtp-simulcast]

Westerlund, M., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP Sessions", draft-westerlund-avtcore-rtp-simulcast-02 (work in progress), February 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.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

8.2. 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.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, 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.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.
- [iana.rtp-pt]
IANA, "RTP Payload types (PT) for standard audio and video encodings", July 2013.

Available at <http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-1>
- [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

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

Email: justin@uberti.name

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

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

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2013

D. Wing
T. Reddy
P. Patil
P. Martinsen
Cisco
June 24, 2013

Mobility with ICE (MICE)
draft-wing-mmusic-ice-mobility-04

Abstract

This specification describes how endpoint mobility can be achieved using ICE. Two mechanisms are shown, one where both endpoints support ICE and another where only one endpoint supports ICE.

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 December 26, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Notational Conventions	4
3. Break Before Make	4
3.1. Absence of other interfaces in Valid list	5
3.1.1. Receiving ICE Mobility event	6
3.2. Keeping unused relayed candidates active	7
3.3. New STUN Attributes	8
4. Make Before Break	8
5. Mobility using TURN	8
5.1. Creating an Allocation	9
5.1.1. Sending an Allocate Request	10
5.1.2. Receiving an Allocate Request	10
5.1.3. Receiving an Allocate Success Response	10
5.1.4. Receiving an Allocate Error Response	10
5.2. Refreshing an Allocation	11
5.2.1. Sending a Refresh Request	11
5.2.2. Receiving a Refresh Request	11
5.2.3. Receiving a Refresh Response	12
5.3. New STUN Attribute MOBILITY-TICKET	12
5.4. New STUN Error Response Code	12
6. Comparison to ICE Restart and Trickle ICE	12
6.1. Break Before Make - ICE Restart	12
6.2. Break Before Make - Trickle ICE	13
7. IANA Considerations	14
8. Security Considerations	14
8.1. Considerations for ICE mechanism	14
8.2. Considerations for TURN mechanism	14
9. Acknowledgements	15
10. Change History	15
10.1. Changes from draft-wing-mmusic-ice-mobility-00 to -01	15
10.2. Changes from draft-wing-mmusic-ice-mobility-01 to -02	15
10.3. Changes from draft-wing-mmusic-ice-mobility-02 to -03	15
11. References	15
11.1. Normative References	15
11.2. Informative References	16
Appendix A.	16
A.1. Presence of other interfaces in Valid list	16
A.1.1. Receiving ICE Mobility event	17
A.2. Losing an Interface	17
A.2.1. Keeping unused candidates in the valid list active	18
Authors' Addresses	19

1. Introduction

When moving between networks, an endpoint has to change its IP address. This change breaks upper layer protocols such as TCP and

RTP. Various techniques exist to prevent this breakage, all tied to making the endpoint's IP address static (e.g., Mobile IP, Proxy Mobile IP, LISP). Other techniques exist, which make the upper layer protocol ambivalent to IP address changes (e.g., SCTP). The mechanisms described in this document are in that last category.

ICE [RFC5245] ensures two endpoints have a working media path between them, and is typically used by Internet-connected interactive media systems (e.g., SIP endpoints). ICE does not expect either the local host or the remote host to change their IP addresses. Although ICE does allow an "ICE restart", this is done by sending a re-INVITE which goes over the SIP signaling path. The SIP signaling path is often slower than the media path (which needs to be recovered as quickly as possible), consumes an extra half round trip, and incurs an additional delay if the mobility event forces the endpoint to re-connect with its SIP proxy. When a device changes its IP address, it is necessary for it to re-establish connectivity with its SIP proxy, which can be performed in parallel with the steps described in this document. This document describes how mobility is performed entirely in the media path, without the additional delay of re-establishing SIP connectivity, issuing a new offer/answer, or the complications of multiple SIP offers. This document considers re-establishing bi-directional media the most critical aspect of a successful mobility event, and its efforts are towards meeting that goal.

A TURN [RFC5766] server relays media packets and is used for a variety of purposes, including overcoming NAT and firewall traversal issues and IP address privacy. The existing TURN specification does not allow the client address to change, especially if multiple clients share the same TURN username (e.g., the same credentials are used on multiple devices).

This document proposes two mechanisms to achieve RTP mobility: a mechanism where both endpoints support MICE, and a mechanism where only one endpoint supports MICE. When both endpoints support MICE, ICE itself can be used to provide mobility. When only one endpoint supports ICE, a TURN server provides mobility. Both mobility techniques work across and between network types (e.g., between 3G and wired Internet access), so long as the client can still access the remote ICE peer or TURN server.

Readers are assumed to be familiar with ICE [RFC5245].

2. Notational Conventions

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

This note uses terminology defined in [RFC5245], and the following additional terminology:

Break Before Make: The initially selected interface for communication may become unavailable (e.g due to loss of coverage when moving out of a WiFi hotspot) and new interfaces may become available due to administrative action (e.g manual activation of a specific connectivity technology) or due to dynamic conditions (e.g. Entering coverage area of a wireless network).

Make Before Break: The initially selected interface for communication may become deprioritized (e.g new interface becoming available and it's per bit cost is cheaper and the connection speed is faster than existing interface used for communication).

Simultaneous Mobility: If both the endpoints are mobile and roam at the same time between networks.

3. Break Before Make

When both endpoints support ICE, ICE itself can provide mobility functions. One of the primary aspects of ICE is its address gathering, wherein ICE has each endpoint determine all of the IP addresses and ports that might be usable for that endpoint and communicate that list of addresses and ports to its peer, usually over SDP. That enables the next primary aspect of ICE, which is its connectivity checks: each ICE endpoint sends a connectivity check from a checklist created by the local and remote candidates exchanged in the initial offer/answer exchange. When the ICE endpoint checks the mapped address from the STUN response during ICE connectivity checks and finds that the transport address does not match any of the local candidates that the ICE agent knows about, the mapped address represents a new candidate -- a peer reflexive candidate. This will cause the endpoint to construct a new pair and insert it into the local checklist (Section 7.2.1.3 of [RFC5245]). ICE Mobility (MICE) takes advantage of that existing ICE functionality to provide faster mobility.

Endpoints that support ICE Mobility perform ICE normally, and MUST also include the MOBILITY-SUPPORT attribute in all of their STUN requests and their STUN responses. The inclusion of this attribute allows the ICE peer to determine if it can achieve mobility using ICE

or needs to use TURN. To force the use of TURN to achieve ICE mobility, the ICE endpoint SHOULD NOT respond to ICE connectivity checks that have an IP address and port different from the TURN server, unless those connectivity checks contain the MOBILITY-SUPPORT attribute. In this way, the remote peer will think those other candidates are invalid (because its connectivity checks did not succeed).

After concluding ICE and moving to the ICE completed state (see Section 8 of [RFC5245] either endpoint or both endpoints can initiate ICE Mobility, no matter if it was the Controlling Agent or the Controlled Agent during normal ICE processing.

3.1. Absence of other interfaces in Valid list

When the interface currently being used for communication becomes unavailable then ICE agent acquires a list of interfaces that are available and based on the locally configured host policy preferences, the ICE endpoint performs ICE Mobility using one of the available interfaces. In this case local candidates from the selected interface are not present in the valid list. ICE Mobility is performed by:

1. The ICE agent remembers the remote host/server reflexive/peer reflexive candidates for each component of the media streams previously used from the valid list before clearing its ICE check list and ICE Valid List.
2. The ICE endpoint gathers host candidates of the same address family as the remote peer on the new interface, forms a check list by creating candidate pairs with local host candidates and remote host/server-reflexive candidates collected in step 1, performs "Computing Pair Priority and Ordering Pairs" (Section 5.7.2 of [RFC5245]), "Pruning the Pairs" (Section 5.7.3 of [RFC5245]), "Computing states" (Section 5.7.4 of [RFC5245]).
3. The ICE endpoint initiates ICE connectivity checks on those candidates from the check list in the previous step, and includes the MOBILITY-EVENT attribute in those connectivity checks.
4. The ICE endpoint acts as controlling agent and the ICE connectivity check from the previous step SHOULD also include the USE-CANDIDATE attribute to signal an aggressive nomination (see Section 2.6 of [RFC5245]).
5. The ICE endpoint performs "Discovering Peer Reflexive Candidates" (Section 7.1.3.2.1 of [RFC5245]), "Constructing a Valid Pair" (Section 7.1.3.2.2 of [RFC5245]), "Updating Pair States"

(Section 7.1.3.2.3 of [RFC5245]), and "Updating the Nominated Flag" (Section 7.1.3.2.4 of [RFC5245]). When the valid list contains a candidate pair for each component then ICE processing is considered complete for the media stream and ICE agent can start sending media using the nominated candidate pair.

6. Once ICE connectivity checks for all of the media streams are completed, the controlling ICE endpoint follows the procedures in Section 11.1 of [RFC5245], specifically to send updated offer if the candidates in the m and c lines for the media stream (called the DEFAULT CANDIDATES) do not match ICE's SELECTED CANDIDATES (also see Appendix B.9 of [RFC5245]).

The ICE endpoint even after Mobility using ICE is successful can issue an updated offer indicating ICE restart if connectivity checks using higher priority candidate pairs are not successful.

Mobility using ICE could fail in case of Simultaneous Mobility or if the ICE peer is behind NAT that performs Address-Dependent Filtering (see Section 5 of [RFC5245]). Hence the ICE endpoint in parallel will re-establish connection with the SIP proxy. It will then determine whether to initiate ICE restart under the following conditions:

1. After re-establishing connection with the SIP proxy and before sending new offer to initiate ICE restart if Mobility using ICE is successful then stop sending the new offer.
2. After successful negotiation of updated offer/answer to initiate ICE restart, proceed with ICE restart and stop Mobility using ICE if ICE checks are in the Running/Failed states or ICE is partially successful and not yet reached ICE complete state. It's not implementation friendly to have to two checks running in parallel. ICE restart can re-use partial successful ICE connectivity check results from Mobility using ICE if required as optimization.

3.1.1. Receiving ICE Mobility event

A STUN Binding Request containing the MOBILITY-EVENT attribute MAY be received by an ICE endpoint. The agent MUST use short-term credential to authenticate the STUN request containing the MOBILITY-EVENT attribute and perform a message integrity check. The ICE endpoint will generate STUN Binding Response containing the MOBILE-SUPPORT attribute and the ICE agent takes role of controlled agent. If STUN Request containing the MOBILITY-EVENT attribute is received before the endpoint is in the ICE Completed state, it should be silently discarded.

The agent remembers the highest-priority nominated pairs in the Valid list for each component of the media stream, called the previous selected pairs before removing all the selected candidate pairs from the Valid List . It continues sending media to that address until it finishes with the steps described below. Because those packets might not be received due to the mobility event, it MAY cache a copy of those packets.

1. The ICE endpoint constructs a pair whose local candidate is equal to the transport address on which the STUN request was received with MOBILITY-EVENT, USE-CANDIDATE attributes and a remote candidate equal to the source transport address where the STUN request came from.
2. The ICE endpoint will add this pair to the valid list if not already present.
3. The agent sets the nominated flag for that pair in the valid pair to true. ICE processing is considered complete for a media stream if the valid list contains a selected candidate pair for each component and ICE agent can start sending media.

The ICE endpoint will follow Steps 1 to 3 when subsequent STUN Binding Requests are received with MOBILITY-EVENT and USE-CANDIDATE attributes.

3.2. Keeping unused relayed candidates active

The ICE endpoints can maintain the relayed candidates active even when not actively used, so that relayed candidates can be tried if ICE connectivity checks using other candidate types fails. The ICE agent will have to create permissions in the TURN server for the remote relayed candidate IP addresses and perform the following steps:

1. The ICE agent will keep the relayed candidates alive using Refresh transaction, as described in [RFC5766].
2. When the endpoint IP address changes due to mobility, the ICE agent will refresh it's allocation with TURN server using Section 5.2.
3. The ICE agent will pair local and remote relayed candidates for connectivity checks when performing the steps in Section 3.1.
4. If the ICE connectivity check succeeds only with local and remote relayed candidates, it suggests that either other peer is roaming at the same time or is behind Address-Dependent Filtering NAT.

The ICE agent adds the relayed candidate pair to the valid list and marks it as selected. The ICE agent can now send media using the newly selected relayed candidate pair. The Mobile device must re-establish connection with SIP proxy, issue an updated offer indicating ICE restart so that media can be switched to higher-priority candidate pairs.

This approach assists Mobility using ICE to succeed but brings in additional overhead of maintaining relayed candidates. In case of Simultaneous Mobility, host candidates can change for both the endpoints by maintaining relayed candidates and using Section 5 media session can be established using the relayed candidate pair.

3.3. New STUN Attributes

Three new attributes are defined by this section: MOBILITY-EVENT, MOBILITY-SUPPORT.

The MOBILITY-EVENT attribute indicates the sender experienced a mobility event. This attribute has no value, thus the attribute length field MUST always be 0. Rules for sending and interpretation of receiving are described above.

The MOBILITY-SUPPORT attribute indicates the sender supports ICE Mobility, as defined in this document. This attribute has no value, thus the attribute length field MUST always be 0. Rules for sending and interpretation of receiving are described above.

4. Make Before Break

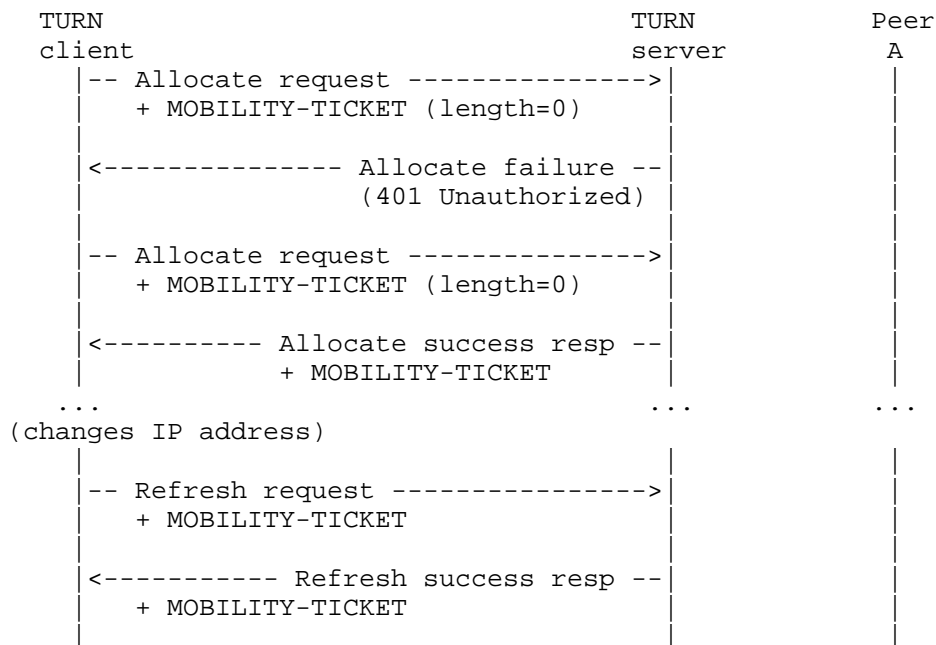
When a new interface comes up and initially selected interface becomes deprioritized (e.g. due to a low cost interface becoming available). The ICE endpoint re-connects to the SIP proxy using the new interface, gathers candidates, exchanges updated offer/exchange to restart ICE. Once ICE processing has reached the Completed state then the ICE endpoint can successfully switch the media over to the new interface. The interface initially used for communication can now be turned off without disrupting communications.

5. Mobility using TURN

To achieve mobility, a TURN client should be able to retain an allocation on the TURN server across changes in the client IP address as a consequence of movement to other networks.

When the client sends the initial Allocate request to the TURN server, it will also include the new STUN attribute MOBILITY-TICKET (with zero length value), which indicates that the client is capable

of mobility and desires a ticket. The TURN server provisions a ticket that is sent inside the new STUN attribute MOBILITY-TICKET in the Allocate Success response to the client. The ticket will be used by the client when it wants to refresh the allocation but with a new client IP address and port. It also ensures that the allocation can only be refreshed this way by the same client. When a client's IP address changes due to mobility, it presents the previously obtained ticket in a Refresh Request to the TURN server. If the ticket is found to be valid, the TURN server will retain the same relayed address/port for the new IP address/port allowing the client to continue using previous channel bindings -- thus, the TURN client does not need to obtain new channel bindings. Any data from external peer will be delivered by the TURN server to this new IP address/port of the client. The TURN client will continue to send application data to its peers using the previously allocated channelBind Requests.



5.1. Creating an Allocation

5.1.1. Sending an Allocate Request

In addition to the process described in Section 6.1 of [RFC5766], the client includes the MOBILITY-TICKET attribute with length 0. This indicates the client is a mobile node and wants a ticket.

5.1.2. Receiving an Allocate Request

In addition to the process described in Section 6.2 of [RFC5766], the server does the following:

If the MOBILITY-TICKET attribute is included, and has length zero, and the TURN session mobility is forbidden by local policy, the server MUST reject the request with the new Mobility Forbidden error code. Following the rules specified in [RFC5389], if the server does not understand the MOBILITY-TICKET attribute, it ignores the attribute.

If the server can successfully process the request create an allocation, the server replies with a success response that includes a STUN MOBILITY-TICKET attribute. TURN server stores it's session state, such as 5-tuple and NONCE, into a ticket that is encrypted by a key known only to the TURN server and sends the ticket in the STUN MOBILITY-TICKET attribute as part of Allocate success response.

The ticket is opaque to the client, so the structure is not subject to interoperability concerns, and implementations may diverge from this format. TURN Allocation state information is encrypted using 128-bit key for Advance Encryption Standard (AES) and 256-bit key for HMAC-SHA-256 for integrity protection.

5.1.3. Receiving an Allocate Success Response

In addition to the process described in Section 6.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the response. This attribute will be presented by the client to the server during a subsequent Refresh request to aid mobility.

5.1.4. Receiving an Allocate Error Response

If the client receives an Allocate error response with error code TBD (Mobility Forbidden), the error is processed as follows:

o TBD (Mobility Forbidden): The request is valid, but the server is refusing to perform it, likely due to administrative restrictions. The client considers the current transaction as having failed. The client MAY notify the user or operator and SHOULD NOT retry the same request with this server until it believes the problem has been fixed.

All other error responses must be handled as described in [RFC5766].

5.2. Refreshing an Allocation

5.2.1. Sending a Refresh Request

If a client wants to refresh an existing allocation and update its time-to-expiry or delete an existing allocation, it will send a Refresh Request as described in Section 7.1 of [RFC5766]. If the client wants to retain the existing allocation in case of IP change, it will include the MOBILITY-TICKET attribute received in the Allocate Success response. If a Refresh transaction was previously made, the MOBILITY-TICKET attribute received in the Refresh Success response of the transaction must be used.

5.2.2. Receiving a Refresh Request

In addition to the process described in Section 7.2 of [RFC5766], the client does the following:

If the STUN MOBILITY-TICKET attribute is included in the Refresh Request then the server will not retrieve the 5-tuple from the packet to identify an associated allocation. Instead TURN server will decrypt the received ticket, verify the ticket's validity and retrieve the 5-tuple allocation from the contents of the ticket. If this 5-tuple obtained from the ticket does not identify an existing allocation then the server MUST reject the request with an error.

If the source IP address and port of the Refresh Request is different from the stored 5-tuple allocation, the TURN server proceeds with checks to see if NONCE in the Refresh request is the same as the one provided in the ticket. The TURN server also uses MESSAGE-INTEGRITY validation to identify the that it is the same user which had previously created the TURN allocation. If the above checks are not successful then server MUST reject the request with a 441 (Wrong Credentials) error.

If all of the above checks pass, the TURN server understands that the client has moved to a new network and acquired a new IP address. The source IP address of the request could either be the host transport address or server-reflexive transport address. The server then

updates it's 5-tuple with the new client IP address and port. TURN server calculates the ticket with the new 5-tuple and sends the new ticket in the STUN MOBILITY-TICKET attribute as part of Refresh Success response.

5.2.3. Receiving a Refresh Response

In addition to the process described in Section 7.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the response. This attribute will be presented by the client to the server during a subsequent Refresh Request to aid mobility.

5.3. New STUN Attribute MOBILITY-TICKET

This attribute is used to retain an Allocation on the TURN server. It is exchanged between the client and server to aid mobility. The value is encrypted and identifies session state such as 5-tuple and NONCE. The value of MOBILITY-TICKET is a variable-length value.

5.4. New STUN Error Response Code

This document defines the following new error response code:

Mobility Forbidden: Mobility request was valid but cannot be performed due to administrative or similar restrictions.

6. Comparison to ICE Restart and Trickle ICE

There has been some concern that ICE Mobility is unnecessary, and that an ICE restart (section 9.1.1.1 of [RFC5245]) would provide exactly the same functionality as ICE Mobility. These sections examine how ICE restart and Trickle ICE [I-D.rescorla-mmusic-ice-trickle] compare with ICE Mobility.

6.1. Break Before Make - ICE Restart

- o If ICE Restart is used for RTP Mobility then in case of Break before Make,
 - 1. Before the endpoint can send an ICE restart message, it has to first re-establish communication with its SIP proxy. This consumes one round-trip for both TCP and UDP. If the connection is protected with TLS (TCP) or DTLS (UDP), we can assume TLS session resumption [RFC5077] will be used to reduce the number of TLS messages. With TLS session resumption, this consumes 1 round trip. If TLS session resumption is not available, a full TLS handshake consumes 2 round trips. This

is a total of 2 round trips (with session resumption) to 3 round trips (without session resumption), which is multiplied by the round trip time to the SIP proxy. The round trip time is dependent on a particular network or deployment, for example in second (2.5G), third (3G) generation wireless networks and satellite communication round trip time could be higher than 250ms. These calculations are only considering the network round-trip time and do not consider the wall-clock time to validate the TLS certificates or generate the TLS keys on the TLS client or the TLS server, which would make this longer.

2. While performing the above steps to re-establish SIP connectivity with its SIP proxy, the endpoint will gather host candidates which incur no network traffic, server-reflexive candidates which incur a round-trip to a STUN server, and relayed candidates which incurs three round trips (two for re-authentication and one for creating the TURN permission). The STUN and TURN communications can be performed in parallel with the SIP connectivity check from step (1), above.
3. The endpoints through the SIP server will exchange offer/answer. The SIP server could also be located halfway around the world from the endpoints and the delay could be significant. For SIP over UDP the endpoint will have send a SIP request and wait for the response to arrive.
4. ICE restart requires sending a new INVITE. A new INVITE cannot be sent if there is an open SIP dialog, such as a previous INVITE. This means rapid mobility events will not work well, and there is also an increased likelihood for glare (both endpoints sending INVITES at the same time).

6.2. Break Before Make - Trickle ICE

- o If Trickle ICE [I-D.rescorla-mmusic-ice-trickle] is used for RTP Mobility then in case of Break before Make,
 1. Trickle ICE can begin connectivity checks while the endpoint is still gathering candidates and can considerably shorten the time necessary for ICE processing to complete. It still involves the overhead of step 1 explained in section Section 6.1.
 2. The endpoint would learn host candidates and inform them to the remote peer in offer, the remote peer will provide its candidates in answer. The host, server reflexive, peer reflexive and relayed candidates of the remote peer may not

change and the remote peer does not have to gather the candidates again. Trickle ICE will test local host candidates with all types of remote candidates provided by the remote peer in the answer. If the ICE peer is behind NAT performing end-point independent mapping and filtering then ICE connectivity checks initiated by the endpoint to the remote peer will succeed.

3. Trickle ICE must be supported by both endpoints for it be used.

7. IANA Considerations

IANA is requested to add the following attributes to the STUN attribute registry [iana-stun],

- o MOBILITY-TICKET (0x802E, in the comprehension-optional range)
- o MOBILITY-EVENT (0x802, in the comprehension-required range)
- o MOBILITY-SUPPORT (0x8000, in the comprehension-optional range)

and to add a new STUN error code "Mobility Forbidden" with the value 501 to the STUN Error Codes registry [iana-stun].

8. Security Considerations

8.1. Considerations for ICE mechanism

A mobility event only occurs after both ICE endpoints have exchanged their ICE information. Thus, both username fragments are already known to both endpoints. Each endpoint contributes at least 24 bits of randomness to the ice-ufrag (Section 15.4 of [RFC5245]), which provides 48 bits of randomness. An off-path attacker would have to guess those 48 bits to cause the endpoints to perform HMAC-SHA1 validation of the MESSAGE-INTEGRITY attribute.

An attacker on the path between the ICE endpoints will see both ice-ufrags, and can cause the endpoints to perform HMAC-SHA1 validation by sending messages from any IP address.

8.2. Considerations for TURN mechanism

TURN server MUST use strong encryption and integrity protection for the ticket to prevent an attacker from using a brute force mechanism to obtain the ticket's contents or refreshing allocations.

Security considerations described in [RFC5766] are also applicable to this mechanism.

9. Acknowledgements

Thanks to Alfred Heggstad, Lishitao, Sujing Zhou, Martin Thomson, Emil Ivov for review and comments.

10. Change History

[Note to RFC Editor: Please remove this section prior to publication.]

10.1. Changes from draft-wing-mmusic-ice-mobility-00 to -01

- o Updated section 3

10.2. Changes from draft-wing-mmusic-ice-mobility-01 to -02

- o Updated Introduction, Notational Conventions, sections 3.1, 3.2.
- o Updated section 3.5

10.3. Changes from draft-wing-mmusic-ice-mobility-02 to -03

- o Moved sections Presence of other interfaces in Valid list, Losing an Interface to Appendix.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

11.2. Informative References

- [I-D.rescorla-mmusic-ice-trickle]
Rescorla, E., Uberti, J., and E. Ivov, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", draft-rescorla-mmusic-ice-trickle-01 (work in progress), October 2012.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.
- [RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.
- [RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, May 2010.
- [RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.
- [iana-stun]
IANA, ., "IANA: STUN Attributes", April 2011, <<http://www.iana.org/assignments/stun-parameters/stun-parameters.xml>>.

Appendix A.

A.1. Presence of other interfaces in Valid list

This technique is optional and only relevant if there is a host policy to maintain unused candidates on other interfaces using the steps in Appendix A.2.1. ICE Agent can maintain unused candidates on other interfaces if it detects that it is behind Address-Dependent Filtering NAT or Firewall. ICE Agent can detect NAT, Firewall behaviour using the procedure explained in [RFC5780]. When the interface currently being used for media communication becomes unavailable. If other interfaces are available and local candidates from these interfaces are already present in the valid list then ICE endpoint will perform the following steps:

1. The ICE endpoint based on the locally configured host policy preferences, will select a interface whose candidates are already present in the valid list.
2. The ICE endpoint clears all the pairs in the valid list containing the IP addresses from the interface that become unavailable.
3. The ICE endpoint initiates ICE connectivity checks on the selected interface. The ICE endpoint acts as controlling agent and MUST include MOBILITY-EVENT attribute to signal mobility event and SHOULD also include the USE-CANDIDATE attribute to signal an aggressive nomination (see Section 2.6 of [RFC5245]). When all components have a nominated pair in the valid list, media can begin to flow using the highest priority nominated pair.
4. The ICE endpoint will re-establish connection with the SIP proxy. Once ICE connectivity checks for all of the media streams are completed, the controlling ICE endpoint follows the procedures in Section 11.1 of [RFC5245], specifically to send updated offer if the candidates in the m and c lines for the media stream (called the DEFAULT CANDIDATES) do not match ICE's SELECTED CANDIDATES (also see Appendix B.9 of [RFC5245]).

The ICE endpoint after Mobility using ICE is successful can issue an updated offer indicating ICE restart if higher priority interface becomes available.

A.1.1. Receiving ICE Mobility event

The ICE endpoint that receives ICE Mobility Event will perform the steps in Section 3.1.1.

A.2. Losing an Interface

When an interface is lost, the SDP MAY be updated, so that the remote ICE host does not waste its efforts with connectivity checks to that address, as those checks will fail. Because it can be argued that this is merely an optimization, and that the interface loss might be temporary (and soon regained), and that ICE has reasonable accommodation for candidates where connectivity checks timeout, this specification does not strongly encourage updating the SDP to remove a lost interface.

Likewise, this specification recommends that ICE candidate addresses in valid list be maintained actively, subject to the host's policy. For example, battery operated hosts have a strong incentive to not

maintain NAT binding for server reflexive candidates learnt through STUN Binding Request, as the maintenance requires sending periodic STUN Binding Indication. As another example, a host that is receiving media over IPv6 may not want to persist with keeping a NATted IPv4 mapping alive (because that consumes a NAT mapping that could be more useful to a host actively utilizing the mapping for real traffic).

Note: this differs from Section 8.3 of [RFC5245], which encourages abandoning unused candidates.

A.2.1. Keeping unused candidates in the valid list active

ICE endpoint subject to host policy can continue performing ICE connectivity checks using candidates from other interfaces on the host even after ICE is complete. If valid list contains unused candidate pairs from other interfaces and one of these interfaces can be selected to send to media in case the existing interface used for media is unavailable then ICE endpoint can keep the unused candidate pairs from other interface{s} alive by sending keepalives every NN seconds. It is recommended to only keep host/server-reflexive candidates active in the valid list and not the relayed candidates.

A.2.1.1. Sending keep alive requests

Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows [RFC6263] describes various reasons for doing keepalives on inactive streams and how to keep NAT mapping alive. However this specification requires some additional functionality associated with the keepalives.

STUN binding requests MUST be used as the keepalive message instead of the STUN Binding indication as specified in [RFC5245]. This is to ensure positive peer consent from the remote side that the candidate pair is still active and in future mobility can be achieved using the steps in Appendix A.1. The request must include the MOBILITY-SUPPORT attribute. If the STUN binding response matches a pair in the checklist then that candidate pair should be kept in the list. If the STUN transaction fails then the candidate pair will be removed from valid list.

A.2.1.2. Receiving keep alive requests

Upon receiving a STUN binding request containing a MOBILITY-SUPPORT attribute even when ICE processing is in the Completed state, the ICE endpoint will add this pair to the valid list if not already present and generate STUN Binding Response containing the MOBILE-SUPPORT attribute.

Authors' Addresses

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tiredy@cisco.com

Prashanth Patil
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marthalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: praspati@cisco.com

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens vei 22
Lysaker, Akershus 1325
Norway

Email: palmarti@cisco.com