

MMUSIC WG
Internet-Draft
Intended status: Informational
Expires: March 19, 2014

R. Even
Huawei Technologies
J. Lennox
Vidyo
Q. Wu
Huawei Technologies
September 15, 2013

The Session Description Protocol (SDP) Application Token Attribute
draft-even-mmusic-application-token-01.txt

Abstract

The RTP fixed header includes the payload type number and the SSRC values of the RTP stream. RTP defines how to 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 March 19, 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	5
3. Proposal for Application ID	5
3.1. appID token	6
3.1.1. RTCP SDES message	8
3.1.2. RTP Header Extension	8
3.1.3. recv-appID	9
4. Using Application ID token in Offer / Answer	10
5. Acknowledgements	13
6. IANA Considerations	13
7. Security Considerations	14
8. References	14
8.1. Normative References	14
8.2. Informative References	14
Authors' Addresses	15

1. Introduction

The RTP [RFC3550] header includes the payload type number and the SSRC values of the RTP stream. RTP defines how to 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.

SDP [RFC4566] can be used to describe multiple RTP media streams in one or more m-lines that define a single SSRC multiplexed RTP session (as specified in [RFC3550]). This addresses the WebRTC architecture [I-D.ietf-rtcweb-overview].

A Unified Plan for Using SDP with Large Numbers of Media Flows [I-D.roach-mmusic-unified-plan] proposes that each m-line will represent a media source [I-D.lennox-raiarea-rtp-grouping-taxonomy]. In the simple case a media source will be one video or audio RTP stream. Media source description becomes more complicated when for robust applications, techniques like RTX and FEC are used to protect media. Also simulcast/layered coding can be used to provide support to heterogeneous receivers. In these cases a media source may send more than one RTP stream, for example, a video stream and a FEC stream.

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 RTP streams described by the SDP m-line.

There is a need to have a token that will allow the mapping between a single RTP streams (identified by an SSRC) in an m-line to the application logic. For example, SSRC1 is the RTP stream from the left camera and SSRC2 is the RTP stream from the right camera specified and SSRC3 is the FEC stream that protect both streams. Note that there are cases where the SSRCs of the RTP streams are not known or may change during the call..

Support of FEC, SVC and simulcast bring more requirements as explained using the following examples.

The first example is of a unified plan [I-D.roach-mmusic-unified-plan] offer of one audio source and one video source. The video source includes two SVC RTP streams a base layer and an enhancement layer. There are also two FEC options:

>Base layer S1 is protected by FEC repair stream R1

Base Layer S1 and Enhancement S2 layers protected by FEC repair stream R2.

This enables the answer to select the base layer with R1 or the Base + enhancement layers both protected by R2.

SDP Offer:

v=0

o=- 20518 0 IN IP4 198.51.100.1

s=FEC Grouping Semantics for SSRC Multiplexing

t=0 0

c=IN IP4 203.0.113.1

a=group:BUNDLE m1 m2

m=audio 56600 RTP/SAVPF 0 109

a=msid:ma ta

a=mid:m1

a=ssrc:53280

a=rtpmap:0 PCMU/8000

a=rtpmap:109 opus/48000

m=video 56602 RTP/AVPF 100 101 110 111 - Main camera

a=msid:ma tb

a=mid:m2

a=rtpmap:100 H264/90000 - Base layer

a=rtpmap:101 H264-SVC/90000 - Enhancement layer.

a=depend:101 lay L1:100 - dependencies

a=rtpmap:110 ld-interleaved-parityfec/90000

```
a=fmtp:110 L=5; D=10; repair-window=200000
a=rtpmap:111 ld-interleaved-parityfec/90000
a=fmtp:111 L=10; D=10; repair-window=400000
a=ssrc:1000 cname:MSTFEC@example.com
a=ssrc:1010 cname:MSTFEC@example.com
a=ssrc:2110 cname:MSTFEC@example.com
a=ssrc:2120 cname:MSTFEC@example.com
a=ssrc-group:FEC-FR 1000 2110
a=ssrc-group:FEC-FR 1000 1010 2120
a=ssrc-group:DDP 1000 1010
```

In this case all video streams are from the same source and can be described using a single m-line. The grouping relations are specified using the SSRCS values that need to be available in the offer. It is also not clear based on the offer which SSRC is mapped to each of the PT numbers.

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 Application ID

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 or session level rather than stream-level. In order to avoid overload of existing attributes, it is better to have a new token attribute that can identify a specific RTP stream corresponding to the application. This document defines such new token, "AppID".

[I-D.roach-mmusic-unified-plan] describes a use case where for early media it is important that the offer will include a token allowing the media receiver to associate it with the correct m-line. This requires that the appID will be the token of the received RTP stream to be used by the sending side. On the other hand to specify the

appIDs of the source RTP stream and the protecting RTP streams there may be a need to specify the sent appID since the relations between the source and repair streams are for the send side and the protection may not be symmetrical. Similar issue may exist for the simulcast use case. This requires having a second optional attribute for the recv-appID to be used for early media.

3.1. appID token

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 also be mapped, for example, to a FEC stream, 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 supports an RTP session as described using SDP. The RTP session may use Bundle [I-D.ietf-mmusic-sdp-bundle-negotiation] with more than one m-lines. In this 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 media receiver in order to map each SSRC to a specific m-line configuration.

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] or a FEC stream that protects multiple RTP streams.

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:" token [SP attribute]

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

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

Examples:

The SSRC of the stream is not known when the SDP offer is sent, an appID is 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
```

The second example is when the application usage of the RTP steam is specified using SDP to provide different image resolutions. The media receiver can map the received SSRC to the specific resolution based on the appId.

Note: This example is using a separate m-line for each offered resolution on the send direction grouped using SCR option [I-D.westerlund-avtcore-rtp-simulcast] It uses the same msid for all grouped image attribute. Other options will be added based on the work done on [I-D.westerlund-avtcore-rtp-simulcast]

```
a=group:SCR 1 2

m=video 49200 RTP/AVP 98

a=rtpmap:98 H264/90000
```

```

imageattr:98 send [x=640,y=360] recv[[x=640,y=360] [x=320,y=180]

a=msid:ma ta
a=appID:2
a=mid:1
m=video 49200 RTP/AVP 99
a=rtpmap:99 H264/90000
imageattr:99 send [x=320,y=180]

a=msid:ma ta
a=appID:3
a=mid:2
a=sendonly

```

3.1.1.1. 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.

3.1.1.2. 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-hdrex: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.

3.1.3. recv-appID

An offer may include a recv-appID attribute allowing the offerer to request from the answerer to use this token for the RTP stream sent from the answerer for a sendrecv or recvonly RTP stream. This is important in order to support early media from the answerer that may be received by the offerer before the answer SDP arrives.

The formal representation of the appID token is:

```

appid-attribute = "recv-appID:" token

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

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

```

4. Using Application ID token in Offer / Answer

The appId may be used in offer answer. Some use cases are provided. They only show part of the SDP that can demonstrate the usage.

the simple case is when each media source describes one RTP stream. In this case the SSRC may be used for the mapping if known but having appId address the case where the SSRC changes. The recv-appID is offered to allow for early media synchronization.

The offer is:

```
m=video 49200 RTP/AVP 99
a=rtpmap:99 H264/90000
a=appId 2
a=recv-appId 10
a=ssrc:20010 CNAME:v1@example.com
m=video 49200 RTP/AVP 100
a=rtpmap:100 H264/90000
a=appId 3
a=recv-appId 20
a=ssrc:20010 CNAME:v2@example.com
```

In this example a three camera system sending three RTP streams protected by a single FEC stream. (note that the full offer may also include a FEC stream for each of the three RTP streams and the answerer may choose which FEC scheme he prefers).

This is the SDP offer for the video sources:

```
v=0
o=- 20518 0 IN IP4 198.51.100.1
s=FEC Grouping Semantics for SSRC Multiplexing
t=0 0
c=IN IP4 203.0.113.1
```

```
a=group:BUNDLE m1 m2 m3 m4 R1
a=group:FEC-FR m2 m3 m4 R1
m=audio 56600 RTP/SAVPF 109
a=mid:m1
a=msid:ma ta
a=appID 1
a=ssrc:53280
a=rtpmap:109 opus/48000
m=video 56602 RTP/AVPF 100 - left camera
a=mid:m2
a=msid:ma tb
a=appID 2
a=rtpmap:100 H264/90000
a=ssrc:1000 cname:MSTFEC@example.com
m=video 56602 RTP/AVPF 101- Middle camera
a=mid:m3
a=msid:ma tc
a=appID 3
a=rtpmap:101 H264/90000
a=ssrc:1010 cname:MSTFEC@example.com
m=video 56602 RTP/AVPF 102 - Right camera
a=mid:m4
a=msid:ma td
a=appID 4
```

```
a=rtpmap:102 H264/90000
a=ssrc:1020 cname:MSTFEC@example.com
m=video 56602 RTP/AVP 110
a=rtpmap:110 ld-interleaved-parityfec/90000
a=fmtp:110 L=5; D=10; repair-window=200000
a=mid:R1
a=appID 5
```

The FEC stream is specified in a separate SDP m-line even though it is not a media source but it does not have any msid so it is not a media stream track. The appID is used to identify this stream as the FEC stream

In the CLUE WG case the mapping is from a media source represented by an SDP m-line to a CLUE Capture encoding specified in the CLUE framework [I-D.ietf-clue-framework]. The mapping may be done using the label attribute.

Example of an offer that offers three CLUE individual encodes. The CLUE config message can be used to map an individual encode to a CLUE media capture [I-D.kyzivat-clue-signaling]. The label value is used in the CLUE protocol to identify CLUE individual encodes. The appId is used to identify the stream by the receiver:

```
a=group:CLUE 4 5 6
...
m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=sendrecv
a=mid:2
a=appID 9
...
```

```
m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:4
a=appID 8
a=label:encl

m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:5
a=appID 7
a=label:enc2

m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:6
a=appID 6
a=label:enc3
```

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.kyzivat-clue-signaling]
Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-kyzivat-clue-signaling-04 (work in progress), July 2013.

- [I-D.lennox-raiarea-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro,
"A Taxonomy of Grouping Semantics and Mechanisms for Real-
Time Transport Protocol (RTP) Sources", draft-lennox-
raiarea-rtp-grouping-taxonomy-01 (work in progress), July
2013.
- [I-D.roach-mmusic-unified-plan]
Roach, A., Uberti, J., and M. Thomson, "A Unified Plan for
Using SDP with Large Numbers of Media Flows", draft-roach-
mmusic-unified-plan-00 (work in progress), July 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.
- [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
Internet-Draft
Obsoletes: 5245 (if approved)
Intended status: Standards Track
Expires: January 16, 2014

A. Keranen
Ericsson
J. Rosenberg
jdrosen.net
July 15, 2013

Interactive Connectivity Establishment (ICE): A Protocol for Network
Address Translator (NAT) Traversal for Offer/Answer Protocols
draft-ietf-mmusic-rfc5245bis-00

Abstract

This document 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).

This document obsoletes RFC 5245.

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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	6
2. Overview of ICE	7
2.1. Gathering Candidate Addresses	9
2.2. Connectivity Checks	11
2.3. Sorting Candidates	12
2.4. Frozen Candidates	13
2.5. Security for Checks	14
2.6. Concluding ICE	14
2.7. Lite Implementations	16
2.8. Usages of ICE	16
3. Terminology	16
4. Sending the Initial Offer	19
4.1. Full Implementation Requirements	20
4.1.1. Gathering Candidates	20
4.1.1.1. Host Candidates	20
4.1.1.2. Server Reflexive and Relayed Candidates	21
4.1.1.3. Computing Foundations	22
4.1.1.4. Keeping Candidates Alive	23
4.1.2. Prioritizing Candidates	23
4.1.2.1. Recommended Formula	23
4.1.2.2. Guidelines for Choosing Type and Local Preferences	24
4.1.3. Eliminating Redundant Candidates	25
4.2. Lite Implementation Requirements	25
4.3. Encoding the Offer	26
5. Receiving the Initial Offer	28
5.1. Verifying ICE Support	28
5.2. Determining Role	29
5.3. Gathering Candidates	30
5.4. Prioritizing Candidates	30
5.5. Encoding the Answer	30
5.6. Forming the Check Lists	30
5.6.1. Forming Candidate Pairs	30
5.6.2. Computing Pair Priority and Ordering Pairs	33
5.6.3. Pruning the Pairs	33
5.6.4. Computing States	33
5.7. Scheduling Checks	36
6. Receipt of the Initial Answer	38
6.1. Verifying ICE Support	38
6.2. Determining Role	38
6.3. Forming the Check List	38
6.4. Performing Ordinary Checks	38
7. Performing Connectivity Checks	38
7.1. STUN Client Procedures	39
7.1.1. Creating Permissions for Relayed Candidates	39
7.1.2. Sending the Request	39

7.1.2.1.	PRIORITY and USE-CANDIDATE	39
7.1.2.2.	ICE-CONTROLLED and ICE-CONTROLLING	40
7.1.2.3.	Forming Credentials	40
7.1.2.4.	DiffServ Treatment	40
7.1.3.	Processing the Response	40
7.1.3.1.	Failure Cases	41
7.1.3.2.	Success Cases	41
7.1.3.2.1.	Discovering Peer Reflexive Candidates	42
7.1.3.2.2.	Constructing a Valid Pair	42
7.1.3.2.3.	Updating Pair States	43
7.1.3.2.4.	Updating the Nominated Flag	44
7.1.3.3.	Check List and Timer State Updates	44
7.2.	STUN Server Procedures	45
7.2.1.	Additional Procedures for Full Implementations	46
7.2.1.1.	Detecting and Repairing Role Conflicts	46
7.2.1.2.	Computing Mapped Address	47
7.2.1.3.	Learning Peer Reflexive Candidates	47
7.2.1.4.	Triggered Checks	48
7.2.1.5.	Updating the Nominated Flag	49
7.2.2.	Additional Procedures for Lite Implementations	49
8.	Concluding ICE Processing	50
8.1.	Procedures for Full Implementations	50
8.1.1.	Nominating Pairs	50
8.1.1.1.	Regular Nomination	50
8.1.1.2.	Aggressive Nomination	51
8.1.2.	Updating States	51
8.2.	Procedures for Lite Implementations	53
8.2.1.	Peer Is Full	53
8.2.2.	Peer Is Lite	53
8.3.	Freeing Candidates	54
8.3.1.	Full Implementation Procedures	54
8.3.2.	Lite Implementation Procedures	54
9.	Keepalives	54
10.	Media Handling	55
10.1.	Sending Media	55
10.1.1.	Procedures for Full Implementations	55
10.1.2.	Procedures for Lite Implementations	56
10.1.3.	Procedures for All Implementations	56
10.2.	Receiving Media	57
11.	Extensibility Considerations	57
12.	Setting Ta and RTO	58
12.1.	RTP Media Streams	58
12.2.	Non-RTP Sessions	60
13.	Example	61
14.	Security Considerations	65
14.1.	Attacks on Connectivity Checks	66
14.2.	Attacks on Server Reflexive Address Gathering	68
14.3.	Attacks on Relayed Candidate Gathering	69

14.4. Insider Attacks	69
14.4.1. STUN Amplification Attack	69
15. STUN Extensions	70
15.1. New Attributes	70
15.2. New Error Response Codes	71
16. Operational Considerations	71
16.1. NAT and Firewall Types	71
16.2. Bandwidth Requirements	71
16.2.1. STUN and TURN Server Capacity Planning	72
16.2.2. Gathering and Connectivity Checks	72
16.2.3. Keepalives	73
16.3. ICE and ICE-lite	73
16.4. Troubleshooting and Performance Management	73
16.5. Endpoint Configuration	74
17. IANA Considerations	74
17.1. STUN Attributes	74
17.2. STUN Error Responses	74
18. IAB Considerations	74
18.1. Problem Definition	75
18.2. Exit Strategy	75
18.3. Brittleness Introduced by ICE	76
18.4. Requirements for a Long-Term Solution	77
18.5. Issues with Existing NAPT Boxes	77
19. Changes from RFC 5245	77
20. Acknowledgements	78
21. References	78
21.1. Normative References	78
21.2. Informative References	78
Appendix A. Lite and Full Implementations	81
Appendix B. Design Motivations	82
B.1. Pacing of STUN Transactions	82
B.2. Candidates with Multiple Bases	83
B.3. Purpose of the Related Address and Related Port Attributes	85
B.4. Importance of the STUN Username	85
B.5. The Candidate Pair Priority Formula	86
B.6. Why Are Keepalives Needed?	87
B.7. Why Prefer Peer Reflexive Candidates?	87
B.8. Why Are Binding Indications Used for Keepalives?	88
Authors' Addresses	88

1. Introduction

RFC 3264 [RFC3264] defines a two-phase exchange of Session Description Protocol (SDP) messages [RFC4566] for the purposes of establishment of multimedia sessions. This offer/answer mechanism is used by protocols such as the Session Initiation Protocol (SIP) [RFC3261].

Protocols using offer/answer are difficult to operate through Network Address Translators (NATs). Because their purpose is to establish a flow of media packets, they tend to carry the IP addresses and ports of media sources and sinks within their messages, which is known to be problematic through NAT [RFC3235]. The protocols also seek to create a media flow directly between participants, so that there is no application layer intermediary between them. This is done to reduce media latency, decrease packet loss, and reduce the operational costs of deploying the application. However, this is difficult to accomplish through NAT. A full treatment of the reasons for this is beyond the scope of this specification.

Numerous solutions have been defined for allowing these protocols to operate through NAT. These include Application Layer Gateways (ALGs), the Middlebox Control Protocol [RFC3303], the original Simple Traversal of UDP Through NAT (STUN) [RFC3489] specification, and Realm Specific IP [RFC3102] [RFC3103] along with session description extensions needed to make them work, such as the Session Description Protocol (SDP) [RFC4566] attribute for the Real Time Control Protocol (RTCP) [RFC3605]. Unfortunately, these techniques all have pros and cons which, make each one optimal in some network topologies, but a poor choice in others. The result is that administrators and implementors are making assumptions about the topologies of the networks in which their solutions will be deployed. This introduces complexity and brittleness into the system. What is needed is a single solution that is flexible enough to work well in all situations.

This specification defines Interactive Connectivity Establishment (ICE) as a technique for NAT traversal for UDP-based media streams (though ICE has been extended to handle other transport protocols, such as TCP [RFC6544]) established by the offer/answer model. ICE is an extension to the offer/answer model, and works by including a multiplicity of IP addresses and ports in the offers and answers, which are then tested for connectivity by peer-to-peer connectivity checks. The IP addresses and ports included in the offer and answer and the connectivity checks are performed using Session Traversal Utilities for NAT (STUN) specification [RFC5389]. ICE also makes use of Traversal Using Relays around NAT (TURN) [RFC5766], an extension to STUN. Because ICE exchanges a multiplicity of IP addresses and

ports for each media stream, it also allows for address selection for multihomed and dual-stack hosts, and for this reason it deprecates [RFC4091] and [RFC4092].

2. Overview of ICE

In a typical ICE deployment, we have two endpoints (known as AGENTS in RFC 3264 terminology) that want to communicate. They are able to communicate indirectly via some signaling protocol (such as SIP), by which they can perform an offer/answer exchange. Note that ICE is not intended for NAT traversal for the signaling protocol, which is assumed to be provided via another mechanism. At the beginning of the ICE process, the agents are ignorant of their own topologies. In particular, they might or might not be behind a NAT (or multiple tiers of NATs). ICE allows the agents to discover enough information about their topologies to potentially find one or more paths by which they can communicate.

Figure 1 shows a typical environment for ICE deployment. The two endpoints are labelled L and R (for left and right, which helps visualize call flows). Both L and R are behind their own respective NATs though they may not be aware of it. The type of NAT and its properties are also unknown. Agents L and R are capable of engaging in an offer/answer exchange, whose purpose is to set up a media session between L and R. Typically, this exchange will occur through a signaling (e.g., SIP) server.

In addition to the agents, a signaling server and NATs, ICE is typically used in concert with STUN or TURN servers in the network. Each agent can have its own STUN or TURN server, or they can be the same.

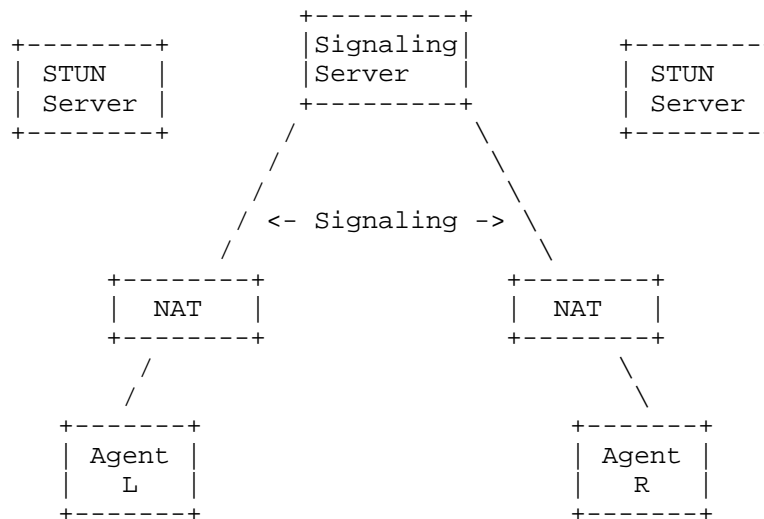


Figure 1: ICE Deployment Scenario

The basic idea behind ICE is as follows: each agent has a variety of candidate TRANSPORT ADDRESSES (combination of IP address and port for a particular transport protocol, which is always UDP in this specification) it could use to communicate with the other agent. These might include:

- o A transport address on a directly attached network interface
- o A translated transport address on the public side of a NAT (a "server reflexive" address)
- o A transport address allocated from a TURN server (a "relayed address")

Potentially, any of L's candidate transport addresses can be used to communicate with any of R's candidate transport addresses. In practice, however, many combinations will not work. For instance, if L and R are both behind NATs, their directly attached interface addresses are unlikely to be able to communicate directly (this is why ICE is needed, after all!). The purpose of ICE is to discover which pairs of addresses will work. The way that ICE does this is to systematically try all possible pairs (in a carefully sorted order) until it finds one or more that work.

2.1. Gathering Candidate Addresses

In order to execute ICE, an agent has to identify all of its address candidates. A CANDIDATE is a transport address -- a combination of IP address and port for a particular transport protocol (with only UDP specified here). This document defines three types of candidates, some derived from physical or logical network interfaces, others discoverable via STUN and TURN. Naturally, one viable candidate is a transport address obtained directly from a local interface. Such a candidate is called a HOST CANDIDATE. The local interface could be Ethernet or WiFi, or it could be one that is obtained through a tunnel mechanism, such as a Virtual Private Network (VPN) or Mobile IP (MIP). In all cases, such a network interface appears to the agent as a local interface from which ports (and thus candidates) can be allocated.

If an agent is multihomed, it obtains a candidate from each IP address. Depending on the location of the PEER (the other agent in the session) on the IP network relative to the agent, the agent may be reachable by the peer through one or more of those IP addresses. Consider, for example, an agent that has a local IP address on a private net 10 network (I1), and a second connected to the public Internet (I2). A candidate from I1 will be directly reachable when communicating with a peer on the same private net 10 network, while a candidate from I2 will be directly reachable when communicating with a peer on the public Internet. Rather than trying to guess which IP address will work prior to sending an offer, the offering agent includes both candidates in its offer.

Next, the agent uses STUN or TURN to obtain additional candidates. These come in two flavors: translated addresses on the public side of a NAT (SERVER REFLEXIVE CANDIDATES) and addresses on TURN servers (RELAYED CANDIDATES). When TURN servers are utilized, both types of candidates are obtained from the TURN server. If only STUN servers are utilized, only server reflexive candidates are obtained from them. The relationship of these candidates to the host candidate is shown in Figure 2. In this figure, both types of candidates are discovered using TURN. In the figure, the notation X:x means IP address X and UDP port x.

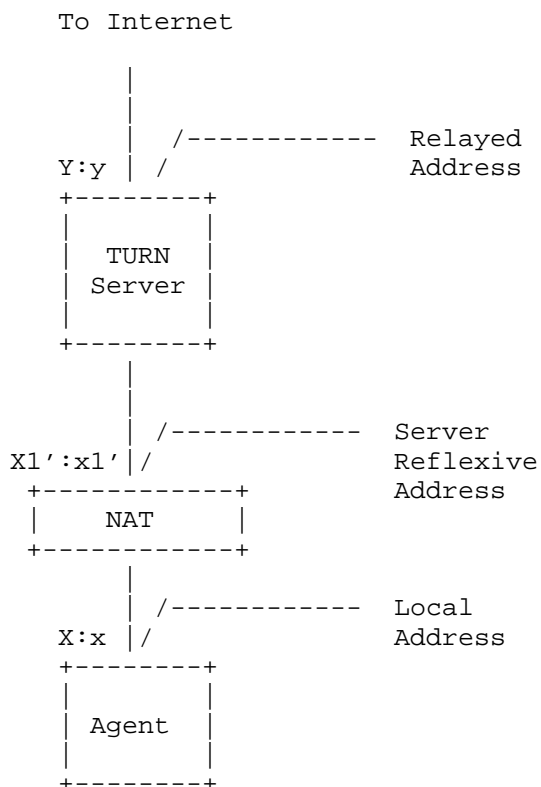


Figure 2: Candidate Relationships

When the agent sends the TURN Allocate request from IP address and port $X:x$, the NAT (assuming there is one) will create a binding $Xl':xl'$, mapping this server reflexive candidate to the host candidate $X:x$. Outgoing packets sent from the host candidate will be translated by the NAT to the server reflexive candidate. Incoming packets sent to the server reflexive candidate will be translated by the NAT to the host candidate and forwarded to the agent. We call the host candidate associated with a given server reflexive candidate the BASE.

Note: "Base" refers to the address an agent sends from for a particular candidate. Thus, as a degenerate case host candidates also have a base, but it's the same as the host candidate.

When there are multiple NATs between the agent and the TURN server, the TURN request will create a binding on each NAT, but only the outermost server reflexive candidate (the one nearest the TURN

server) will be discovered by the agent. If the agent is not behind a NAT, then the base candidate will be the same as the server reflexive candidate and the server reflexive candidate is redundant and will be eliminated.

The Allocate request then arrives at the TURN server. The TURN server allocates a port *y* from its local IP address *Y*, and generates an Allocate response, informing the agent of this relayed candidate. The TURN server also informs the agent of the server reflexive candidate, *Xl':xl'* by copying the source transport address of the Allocate request into the Allocate response. The TURN server acts as a packet relay, forwarding traffic between *L* and *R*. In order to send traffic to *L*, *R* sends traffic to the TURN server at *Y:y*, and the TURN server forwards that to *Xl':xl'*, which passes through the NAT where it is mapped to *X:x* and delivered to *L*.

When only STUN servers are utilized, the agent sends a STUN Binding request [RFC5389] to its STUN server. The STUN server will inform the agent of the server reflexive candidate *Xl':xl'* by copying the source transport address of the Binding request into the Binding response.

2.2. Connectivity Checks

Once *L* has gathered all of its candidates, it orders them in highest to lowest-priority and sends them to *R* over the signaling channel. The candidates are carried in attributes in the offer. When *R* receives the offer, it performs the same gathering process and responds with its own list of candidates. At the end of this process, each agent has a complete list of both its candidates and its peer's candidates. It pairs them up, resulting in CANDIDATE PAIRS. To see which pairs work, each agent schedules a series of CHECKS. Each check is a STUN request/response transaction that the client will perform on a particular candidate pair by sending a STUN request from the local candidate to the remote candidate.

The basic principle of the connectivity checks is simple:

1. Sort the candidate pairs in priority order.
2. Send checks on each candidate pair in priority order.
3. Acknowledge checks received from the other agent.

With both agents performing a check on a candidate pair, the result is a 4-way handshake:

```

L                                     R
-                                     -
STUN request ->                      \ L's
                                     / check
      <- STUN response
                                     \ R's
      <- STUN request                  /
STUN response ->                      / check

```

Figure 3: Basic Connectivity Check

It is important to note that the STUN requests are sent to and from the exact same IP addresses and ports that will be used for media (e.g., RTP and RTCP). Consequently, agents demultiplex STUN and RTP/RTCP using contents of the packets, rather than the port on which they are received. Fortunately, this demultiplexing is easy to do, especially for RTP and RTCP.

Because a STUN Binding request is used for the connectivity check, the STUN Binding response will contain the agent's translated transport address on the public side of any NATs between the agent and its peer. If this transport address is different from other candidates the agent already learned, it represents a new candidate, called a PEER REFLEXIVE CANDIDATE, which then gets tested by ICE just the same as any other candidate.

As an optimization, as soon as R gets L's check message, R schedules a connectivity check message to be sent to L on the same candidate pair. This accelerates the process of finding a valid candidate, and is called a TRIGGERED CHECK.

At the end of this handshake, both L and R know that they can send (and receive) messages end-to-end in both directions.

2.3. Sorting Candidates

Because the algorithm above searches all candidate pairs, if a working pair exists it will eventually find it no matter what order the candidates are tried in. In order to produce faster (and better) results, the candidates are sorted in a specified order. The resulting list of sorted candidate pairs is called the CHECK LIST. The algorithm is described in Section 4.1.2 but follows two general principles:

- o Each agent gives its candidates a numeric priority, which is sent along with the candidate to the peer.
- o The local and remote priorities are combined so that each agent has the same ordering for the candidate pairs.

The second property is important for getting ICE to work when there are NATs in front of L and R. Frequently, NATs will not allow packets in from a host until the agent behind the NAT has sent a packet towards that host. Consequently, ICE checks in each direction will not succeed until both sides have sent a check through their respective NATs.

The agent works through this check list by sending a STUN request for the next candidate pair on the list periodically. These are called ORDINARY CHECKS.

In general, the priority algorithm is designed so that candidates of similar type get similar priorities and so that more direct routes (that is, through fewer media relays and through fewer NATs) are preferred over indirect ones (ones with more media relays and more NATs). Within those guidelines, however, agents have a fair amount of discretion about how to tune their algorithms.

2.4. Frozen Candidates

The previous description only addresses the case where the agents wish to establish a media session with one COMPONENT (a piece of a media stream requiring a single transport address; a media stream may require multiple components, each of which has to work for the media stream as a whole to be work). Often (e.g., with RTP and RTCP), the agents actually need to establish connectivity for more than one flow.

The network properties are likely to be very similar for each component (especially because RTP and RTCP are sent and received from the same IP address). It is usually possible to leverage information from one media component in order to determine the best candidates for another. ICE does this with a mechanism called "frozen candidates".

Each candidate is associated with a property called its FOUNDATION. Two candidates have the same foundation when they are "similar" -- of the same type and obtained from the same host candidate and STUN/TURN server using the same protocol. Otherwise, their foundation is different. A candidate pair has a foundation too, which is just the concatenation of the foundations of its two candidates. Initially, only the candidate pairs with unique foundations are tested. The other candidate pairs are marked "frozen". When the connectivity checks for a candidate pair succeed, the other candidate pairs with the same foundation are unfrozen. This avoids repeated checking of components that are superficially more attractive but in fact are likely to fail.

While we've described "frozen" here as a separate mechanism for expository purposes, in fact it is an integral part of ICE and the ICE prioritization algorithm automatically ensures that the right candidates are unfrozen and checked in the right order. However, if the ICE usage does not utilize multiple components or media streams, it does not need to implement this algorithm.

2.5. Security for Checks

Because ICE is used to discover which addresses can be used to send media between two agents, it is important to ensure that the process cannot be hijacked to send media to the wrong location. Each STUN connectivity check is covered by a message authentication code (MAC) computed using a key exchanged in the signaling channel. This MAC provides message integrity and data origin authentication, thus stopping an attacker from forging or modifying connectivity check messages. Furthermore, if for example a SIP [RFC3261] caller is using ICE, and their call forks, the ICE exchanges happen independently with each forked recipient. In such a case, the keys exchanged in the signaling help associate each ICE exchange with each forked recipient.

2.6. Concluding ICE

ICE checks are performed in a specific sequence, so that high-priority candidate pairs are checked first, followed by lower-priority ones. One way to conclude ICE is to declare victory as soon as a check for each component of each media stream completes successfully. Indeed, this is a reasonable algorithm, and details for it are provided below. However, it is possible that a packet loss will cause a higher-priority check to take longer to complete. In that case, allowing ICE to run a little longer might produce better results. More fundamentally, however, the prioritization defined by this specification may not yield "optimal" results. As an example, if the aim is to select low-latency media paths, usage of a relay is a hint that latencies may be higher, but it is nothing more than a hint. An actual round-trip time (RTT) measurement could be made, and it might demonstrate that a pair with lower priority is actually better than one with higher priority.

Consequently, ICE assigns one of the agents in the role of the CONTROLLING AGENT, and the other of the CONTROLLED AGENT. The controlling agent gets to nominate which candidate pairs will get used for media amongst the ones that are valid. It can do this in one of two ways -- using REGULAR NOMINATION or AGGRESSIVE NOMINATION.

With regular nomination, the controlling agent lets the checks continue until at least one valid candidate pair for each media

stream is found. Then, it picks amongst those that are valid, and sends a second STUN request on its NOMINATED candidate pair, but this time with a flag set to tell the peer that this pair has been nominated for use. This is shown in Figure 4.

```

L                                     R
-                                     -
STUN request ->                      \ L's
      <- STUN response                /  check

      <- STUN request                \ R's
STUN response ->                    /  check

STUN request + flag ->              \ L's
      <- STUN response                /  check

```

Figure 4: Regular Nomination

Once the STUN transaction with the flag completes, both sides cancel any future checks for that media stream. ICE will now send media using this pair. The pair an ICE agent is using for media is called the SELECTED PAIR.

In aggressive nomination, the controlling agent puts the flag in every connectivity check STUN request it sends. This way, once the first check succeeds, ICE processing is complete for that media stream and the controlling agent doesn't have to send a second STUN request. The selected pair will be the highest-priority valid pair whose check succeeded. Aggressive nomination is faster than regular nomination, but gives less flexibility. Aggressive nomination is shown in Figure 5.

```

L                                     R
-                                     -
STUN request + flag ->                \ L's
      <- STUN response                  /  check

      <- STUN request                  \ R's
STUN response ->                      /  check

```

Figure 5: Aggressive Nomination

Once ICE is concluded, it can be restarted at any time for one or all of the media streams by either agent. This is done by sending an

updated offer indicating a restart.

2.7. Lite Implementations

In order for ICE to be used in a call, both agents need to support it. However, certain agents will always be connected to the public Internet and have a public IP address at which it can receive packets from any correspondent. To make it easier for these devices to support ICE, ICE defines a special type of implementation called LITE (in contrast to the normal FULL implementation). A lite implementation doesn't gather candidates; it includes only host candidates for any media stream. Lite agents do not generate connectivity checks or run the state machines, though they need to be able to respond to connectivity checks. When a lite implementation connects with a full implementation, the full agent takes the role of the controlling agent, and the lite agent takes on the controlled role. When two lite implementations connect, no checks are sent.

For guidance on when a lite implementation is appropriate, see the discussion in Appendix A.

It is important to note that the lite implementation was added to this specification to provide a stepping stone to full implementation. Even for devices that are always connected to the public Internet, a full implementation is preferable if achievable.

2.8. Usages of ICE

This document specifies generic use of ICE with protocols that provide offer/answer semantics. The specific details (e.g., how to encode candidates) for different protocols using ICE are described in separate usage documents. For example, usage with SIP and SDP is described in [I-D.petithuguenin-mmusic-ice-sip-sdp].

3. Terminology

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

Readers should be familiar with the terminology defined in the offer/answer model [RFC3264], STUN [RFC5389], and NAT Behavioral requirements for UDP [RFC4787].

This specification makes use of the following additional terminology:

Agent: As defined in RFC 3264, an agent is the protocol implementation involved in the offer/answer exchange. There are two agents involved in an offer/answer exchange.

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

Transport Address: The combination of an IP address and transport protocol (such as UDP or TCP) port.

Media, Media Stream: When ICE is used to setup multimedia sessions, the media is usually transported over RTP, and a media stream composes of a stream of RTP packets. When ICE is used with other than multimedia sessions, the terms "media" and "media stream" are still used in this specification to refer to the IP data packets that are exchanged between the peers on the path created and tested with ICE.

Candidate: A transport address that is a potential point of contact for receipt of media. Candidates also have properties -- their type (server reflexive, relayed, or host), priority, foundation, and base.

Component: A component is a piece of a media stream requiring a single transport address; a media stream may require multiple components, each of which has to work for the media stream as a whole to work. For media streams based on RTP, there are two components per media stream -- one for RTP, and one for RTCP.

Host Candidate: A candidate obtained by binding to a specific port from an IP address on the host. This includes IP addresses on physical interfaces and logical ones, such as ones obtained through Virtual Private Networks (VPNs) and Realm Specific IP (RSIP) [RFC3102] (which lives at the operating system level).

Server Reflexive Candidate: A candidate whose IP address and port are a binding allocated by a NAT for an agent when it sent a packet through the NAT to a server. Server reflexive candidates can be learned by STUN servers using the Binding request, or TURN servers, which provides both a relayed and server reflexive candidate.

Peer Reflexive Candidate: A candidate whose IP address and port are a binding allocated by a NAT for an agent when it sent a STUN Binding request through the NAT to its peer.

Relayed Candidate: A candidate obtained by sending a TURN Allocate request from a host candidate to a TURN server. The relayed candidate is resident on the TURN server, and the TURN server relays packets back towards the agent.

Base: The base of a server reflexive candidate is the host candidate from which it was derived. A host candidate is also said to have a base, equal to that candidate itself. Similarly, the base of a relayed candidate is that candidate itself.

Foundation: An arbitrary string that is the same for two candidates that have the same type, base IP address, protocol (UDP, TCP, etc.), and STUN or TURN server. If any of these are different, then the foundation will be different. Two candidate pairs with the same foundation pairs are likely to have similar network characteristics. Foundations are used in the frozen algorithm.

Local Candidate: A candidate that an agent has obtained and included in an offer or answer it sent.

Remote Candidate: A candidate that an agent received in an offer or answer from its peer.

Default Destination/Candidate: The default destination for a component of a media stream is the transport address that would be used by an agent that is not ICE aware. A default candidate for a component is one whose transport address matches the default destination for that component.

Candidate Pair: A pairing containing a local candidate and a remote candidate.

Check, Connectivity Check, STUN Check: A STUN Binding request transaction for the purposes of verifying connectivity. A check is sent from the local candidate to the remote candidate of a candidate pair.

Check List: An ordered set of candidate pairs that an agent will use to generate checks.

Ordinary Check: A connectivity check generated by an agent as a consequence of a timer that fires periodically, instructing it to send a check.

Triggered Check: A connectivity check generated as a consequence of the receipt of a connectivity check from the peer.

Valid List: An ordered set of candidate pairs for a media stream that have been validated by a successful STUN transaction.

Full: An ICE implementation that performs the complete set of functionality defined by this specification.

Lite: An ICE implementation that omits certain functions, implementing only as much as is necessary for a peer implementation that is full to gain the benefits of ICE. Lite implementations do not maintain any of the state machines and do not generate connectivity checks.

Controlling Agent: The ICE agent that is responsible for selecting the final choice of candidate pairs and signaling them through STUN. In any session, one agent is always controlling. The other is the controlled agent.

Controlled Agent: An ICE agent that waits for the controlling agent to select the final choice of candidate pairs.

Regular Nomination: The process of picking a valid candidate pair for media traffic by validating the pair with one STUN request, and then picking it by sending a second STUN request with a flag indicating its nomination.

Aggressive Nomination: The process of picking a valid candidate pair for media traffic by including a flag in every connectivity check STUN request, such that the first one to produce a valid candidate pair is used for media.

Nominated: If a valid candidate pair has its nominated flag set, it means that it may be selected by ICE for sending and receiving media.

Selected Pair, Selected Candidate: The candidate pair selected by ICE for sending and receiving media is called the selected pair, and each of its candidates is called the selected candidate.

Using Protocol, ICE Usage: The protocol that uses ICE for NAT traversal. A usage specification defines the protocol specific details on how the procedures defined here are applied to that protocol.

4. Sending the Initial Offer

In order to send the initial offer in an offer/answer exchange, an agent must (1) gather candidates, (2) prioritize them, (3) eliminate

redundant candidates, (4) (possibly) choose default candidates, and then (5) formulate and send the offer. All but the last of these five steps differ for full and lite implementations.

4.1. Full Implementation Requirements

4.1.1. Gathering Candidates

An agent gathers candidates when it believes that communication is imminent. An offerer can do this based on a user interface cue, or based on an explicit request to initiate a session. Every candidate is a transport address. It also has a type and a base. Four types are defined and gathered by this specification -- host candidates, server reflexive candidates, peer reflexive candidates, and relayed candidates. The server reflexive candidates are gathered using STUN or TURN, and relayed candidates are obtained through TURN. Peer reflexive candidates are obtained in later phases of ICE, as a consequence of connectivity checks. The base of a candidate is the candidate that an agent must send from when using that candidate.

4.1.1.1. Host Candidates

The first step is to gather host candidates. Host candidates are obtained by binding to ports (typically ephemeral) on a IP address attached to an interface (physical or virtual, including VPN interfaces) on the host.

For each UDP media stream the agent wishes to use, the agent SHOULD obtain a candidate for each component of the media stream on each IP address that the host has, with the exceptions listed below. The agent obtains each candidate by binding to a UDP port on the specific IP address. A host candidate (and indeed every candidate) is always associated with a specific component for which it is a candidate. Each component has an ID assigned to it, called the component ID. For RTP-based media streams, the RTP itself has a component ID of 1, and RTCP a component ID of 2. If an agent is using RTCP, it MUST obtain a candidate for it. If an agent is using both RTP and RTCP, it would end up with 2*K host candidates if an agent has K IP addresses.

The base for each host candidate is set to the candidate itself.

The host candidates are gathered from all IP addresses with the following exceptions:

- o Addresses from a loopback interface MUST NOT be included in the candidate addresses.

- o Deprecated IPv4-compatible IPv6 addresses [RFC4291] and IPv6 site-local unicast addresses [RFC3879] MUST NOT be included in the address candidates.
- o IPv4-mapped IPv6 addresses SHOULD NOT be included in the offered candidates unless the application using ICE does not support IPv4 (i.e., is an IPv6-only application [RFC4038]).

4.1.1.2. Server Reflexive and Relayed Candidates

Agents SHOULD obtain relayed candidates and SHOULD obtain server reflexive candidates. These requirements are at SHOULD strength to allow for provider variation. Use of STUN and TURN servers may be unnecessary in closed networks where agents are never connected to the public Internet or to endpoints outside of the closed network. In such cases, a full implementation would be used for agents that are dual-stack or multihomed, to select a host candidate. Use of TURN servers is expensive, and when ICE is being used, they will only be utilized when both endpoints are behind NATs that perform address and port dependent mapping. Consequently, some deployments might consider this use case to be marginal, and elect not to use TURN servers. If an agent does not gather server reflexive or relayed candidates, it is RECOMMENDED that the functionality be implemented and just disabled through configuration, so that it can be re-enabled through configuration if conditions change in the future.

If an agent is gathering both relayed and server reflexive candidates, it uses a TURN server. If it is gathering just server reflexive candidates, it uses a STUN server.

The agent next pairs each host candidate with the STUN or TURN server with which it is configured or has discovered by some means. If a STUN or TURN server is configured, it is RECOMMENDED that a domain name be configured, and the DNS procedures in [RFC5389] (using SRV records with the "stun" service) be used to discover the STUN server, and the DNS procedures in [RFC5766] (using SRV records with the "turn" service) be used to discover the TURN server.

This specification only considers usage of a single STUN or TURN server. When there are multiple choices for that single STUN or TURN server (when, for example, they are learned through DNS records and multiple results are returned), an agent SHOULD use a single STUN or TURN server (based on its IP address) for all candidates for a particular session. This improves the performance of ICE. The result is a set of pairs of host candidates with STUN or TURN servers. The agent then chooses one pair, and sends a Binding or Allocate request to the server from that host candidate. Binding requests to a STUN server are not authenticated, and any ALTERNATE-

SERVER attribute in a response is ignored. Agents MUST support the backwards compatibility mode for the Binding request defined in [RFC5389]. Allocate requests SHOULD be authenticated using a long-term credential obtained by the client through some other means.

Every T_a milliseconds thereafter, the agent can generate another new STUN or TURN transaction. This transaction can either be a retry of a previous transaction that failed with a recoverable error (such as authentication failure), or a transaction for a new host candidate and STUN or TURN server pair. The agent SHOULD NOT generate transactions more frequently than one every T_a milliseconds. See Section 12 for guidance on how to set T_a and the STUN retransmit timer, RTO .

The agent will receive a Binding or Allocate response. A successful Allocate response will provide the agent with a server reflexive candidate (obtained from the mapped address) and a relayed candidate in the XOR-RELAYED-ADDRESS attribute. If the Allocate request is rejected because the server lacks resources to fulfill it, the agent SHOULD instead send a Binding request to obtain a server reflexive candidate. A Binding response will provide the agent with only a server reflexive candidate (also obtained from the mapped address). The base of the server reflexive candidate is the host candidate from which the Allocate or Binding request was sent. The base of a relayed candidate is that candidate itself. If a relayed candidate is identical to a host candidate (which can happen in rare cases), the relayed candidate MUST be discarded.

4.1.1.3. Computing Foundations

Finally, the agent assigns each candidate a foundation. The foundation is an identifier, scoped within a session. Two candidates MUST have the same foundation ID when all of the following are true:

- o they are of the same type (host, relayed, server reflexive, or peer reflexive)
- o their bases have the same IP address (the ports can be different)
- o for reflexive and relayed candidates, the STUN or TURN servers used to obtain them have the same IP address
- o they were obtained using the same transport protocol (TCP, UDP, etc.)

Similarly, two candidates MUST have different foundations if their types are different, their bases have different IP addresses, the STUN or TURN servers used to obtain them have different IP addresses,

or their transport protocols are different.

4.1.1.4. Keeping Candidates Alive

Once server reflexive and relayed candidates are allocated, they **MUST** be kept alive until ICE processing has completed, as described in Section 8.3. For server reflexive candidates learned through a Binding request, the bindings **MUST** be kept alive by additional Binding requests to the server. Refreshes for allocations are done using the Refresh transaction, as described in [RFC5766]. The Refresh requests will also refresh the server reflexive candidate.

4.1.2. Prioritizing Candidates

The prioritization process results in the assignment of a priority to each candidate. Each candidate for a media stream **MUST** have a unique priority that **MUST** be a positive integer between 1 and $(2^{31} - 1)$. This priority will be used by ICE to determine the order of the connectivity checks and the relative preference for candidates.

An agent **SHOULD** compute this priority using the formula in Section 4.1.2.1 and choose its parameters using the guidelines in Section 4.1.2.2. If an agent elects to use a different formula, ICE will take longer to converge since both agents will not be coordinated in their checks.

4.1.2.1. Recommended Formula

When using the formula, an agent computes the priority by determining a preference for each type of candidate (server reflexive, peer reflexive, relayed, and host), and, when the agent is multihomed, choosing a preference for its IP addresses. These two preferences are then combined to compute the priority for a candidate. That priority is computed using the following formula:

$$\text{priority} = (2^{24}) * (\text{type preference}) + \\ (2^8) * (\text{local preference}) + \\ (2^0) * (256 - \text{component ID})$$

The type preference **MUST** be an integer from 0 to 126 inclusive, and represents the preference for the type of the candidate (where the types are local, server reflexive, peer reflexive, and relayed). A 126 is the highest preference, and a 0 is the lowest. Setting the value to a 0 means that candidates of this type will only be used as a last resort. The type preference **MUST** be identical for all candidates of the same type and **MUST** be different for candidates of

different types. The type preference for peer reflexive candidates MUST be higher than that of server reflexive candidates. Note that candidates gathered based on the procedures of Section 4.1.1 will never be peer reflexive candidates; candidates of these type are learned from the connectivity checks performed by ICE.

The local preference MUST be an integer from 0 to 65535 inclusive. It represents a preference for the particular IP address from which the candidate was obtained, in cases where an agent is multihomed. 65535 represents the highest preference, and a zero, the lowest. When there is only a single IP address, this value SHOULD be set to 65535. More generally, if there are multiple candidates for a particular component for a particular media stream that have the same type, the local preference MUST be unique for each one. In this specification, this only happens for multihomed hosts. If a host is multihomed because it is dual-stack, the local preference SHOULD be set equal to the precedence value for IP addresses described in RFC 6724 [RFC6724]. If the host operating system provides an API for discovering preference among different addresses, those preferences SHOULD be used for the local preference to prioritize addresses indicated as preferred by the operating system.

The component ID is the component ID for the candidate, and MUST be between 1 and 256 inclusive.

4.1.2.2. Guidelines for Choosing Type and Local Preferences

One criterion for selection of the type and local preference values is the use of a media intermediary, such as a TURN server, VPN server, or NAT. With a media intermediary, if media is sent to that candidate, it will first transit the media intermediary before being received. Relayed candidates are one type of candidate that involves a media intermediary. Another are host candidates obtained from a VPN interface. When media is transited through a media intermediary, it can increase the latency between transmission and reception. It can increase the packet losses, because of the additional router hops that may be taken. It may increase the cost of providing service, since media will be routed in and right back out of a media intermediary run by a provider. If these concerns are important, the type preference for relayed candidates SHOULD be lower than host candidates. The RECOMMENDED values are 126 for host candidates, 100 for server reflexive candidates, 110 for peer reflexive candidates, and 0 for relayed candidates. Furthermore, if an agent is multihomed and has multiple IP addresses, the local preference for host candidates from a VPN interface SHOULD have a priority of 0.

Another criterion for selection of preferences is IP address family. ICE works with both IPv4 and IPv6. It therefore provides a

transition mechanism that allows dual-stack hosts to prefer connectivity over IPv6, but to fall back to IPv4 in case the v6 networks are disconnected (due, for example, to a failure in a 6to4 relay) [RFC3056]. It can also help with hosts that have both a native IPv6 address and a 6to4 address. In such a case, higher local preferences could be assigned to the v6 addresses, followed by the 6to4 addresses, followed by the v4 addresses. This allows a site to obtain and begin using native v6 addresses immediately, yet still fall back to 6to4 addresses when communicating with agents in other sites that do not yet have native v6 connectivity.

Another criterion for selecting preferences is security. If a user is a telecommuter, and therefore connected to a corporate network and a local home network, the user may prefer their voice traffic to be routed over the VPN in order to keep it on the corporate network when communicating within the enterprise, but use the local network when communicating with users outside of the enterprise. In such a case, a VPN address would have a higher local preference than any other address.

Another criterion for selecting preferences is topological awareness. This is most useful for candidates that make use of intermediaries. In those cases, if an agent has preconfigured or dynamically discovered knowledge of the topological proximity of the intermediaries to itself, it can use that to assign higher local preferences to candidates obtained from closer intermediaries.

4.1.3. Eliminating Redundant Candidates

Next, the agent eliminates redundant candidates. A candidate is redundant if its transport address equals another candidate, and its base equals the base of that other candidate. Note that two candidates can have the same transport address yet have different bases, and these would not be considered redundant. Frequently, a server reflexive candidate and a host candidate will be redundant when the agent is not behind a NAT. The agent SHOULD eliminate the redundant candidate with the lower priority.

4.2. Lite Implementation Requirements

Lite implementations only utilize host candidates. A lite implementation MUST, for each component of each media stream, allocate zero or one IPv4 candidates. It MAY allocate zero or more IPv6 candidates, but no more than one per each IPv6 address utilized by the host. Since there can be no more than one IPv4 candidate per component of each media stream, if an agent has multiple IPv4 addresses, it MUST choose one for allocating the candidate. If a host is dual-stack, it is RECOMMENDED that it allocate one IPv4

candidate and one global IPv6 address. With the lite implementation, ICE cannot be used to dynamically choose amongst candidates. Therefore, including more than one candidate from a particular scope is NOT RECOMMENDED, since only a connectivity check can truly determine whether to use one address or the other.

Each component has an ID assigned to it, called the component ID. For RTP-based media streams, the RTP itself has a component ID of 1, and RTCP a component ID of 2. If an agent is using RTCP, it MUST obtain candidates for it.

Each candidate is assigned a foundation. The foundation MUST be different for two candidates allocated from different IP addresses, and MUST be the same otherwise. A simple integer that increments for each IP address will suffice. In addition, each candidate MUST be assigned a unique priority amongst all candidates for the same media stream. This priority SHOULD be equal to:

$$\text{priority} = (2^{24}) * (126) + \\ (2^8) * (\text{IP precedence}) + \\ (2^0) * (256 - \text{component ID})$$

If a host is v4-only, it SHOULD set the IP precedence to 65535. If a host is v6 or dual-stack, the IP precedence SHOULD be the precedence value for IP addresses described in RFC 6724 [RFC6724].

Next, an agent chooses a default candidate for each component of each media stream. If a host is IPv4-only, there would only be one candidate for each component of each media stream, and therefore that candidate is the default. If a host is IPv6 or dual-stack, the selection of default is a matter of local policy. This default SHOULD be chosen such that it is the candidate most likely to be used with a peer. For IPv6-only hosts, this would typically be a globally scoped IPv6 address. For dual-stack hosts, the IPv4 address is RECOMMENDED.

4.3. Encoding the Offer

The syntax for the offer and answer messages is entirely a matter of convenience for the using protocol. However, the following parameters and their data types needs to be conveyed in the initial exchange:

Candidate attribute There will be one or more of these for each "media stream". Each candidate is composed of:

Connection Address: The IP address and transport protocol port of the candidate.

Transport: An indicator of the transport protocol for this candidate. This need not be present if the using protocol will only ever run over a single transport protocol. If it runs over more than one, or if others are anticipated to be used in the future, this should be present.

Foundation: A sequence of up to 32 characters.

Component-ID: This would be present only if the using protocol were utilizing the concept of components. If it is, it would be a positive integer that indicates the component ID for which this is a candidate.

Priority: An encoding of the 32-bit priority value.

Candidate Type: The candidate type, as defined in ICE.

Related Address and Port: The related IP address and port for this candidate, as defined by ICE.

Extensibility Parameters: The using protocol should define some means for adding new per-candidate ICE parameters in the future.

Lite Flag: If ICE lite is used by the using protocol, it needs to convey a boolean parameter which indicates whether the implementation is lite or not.

Username Fragment and Password: The using protocol has to convey a username fragment and password. The username fragment MUST contain at least 24 bits of randomness, and the password MUST contain at least 128 bits of randomness.

ICE extensions: In addition to the per-candidate extensions above, the using protocol should allow for new media-stream or session-level attributes (ice-options).

If the using protocol is using the ICE mismatch feature, a way is needed to convey this parameter in answers. It is a boolean flag.

The exchange of parameters is symmetric; both agents need to send the same set of attributes as defined above.

The using protocol may (or may not) need to deal with backwards compatibility with older implementations that do not support ICE. If the fallback mechanism is being used, then presumably the using protocol provides a way of conveying the default candidate (its IP address and port) in addition to the ICE parameters.

STUN connectivity checks between agents are authenticated using the short-term credential mechanism defined for STUN [RFC5389]. This mechanism relies on a username and password that are exchanged through protocol machinery between the client and server. With ICE, the offer/answer exchange is used to exchange them. The username part of this credential is formed by concatenating a username fragment from each agent, separated by a colon. Each agent also provides a password, used to compute the message integrity for requests it receives. The username fragment and password are exchanged in the offer and answer. In addition to providing security, the username provides disambiguation and correlation of checks to media streams. See Appendix B.4 for motivation.

If an agent is a lite implementation, it **MUST** indicate this in the offer.

ICE provides for extensibility by allowing an offer or answer to contain a series of tokens that identify the ICE extensions used by that agent. If an agent supports an ICE extension, it **MUST** include the token defined for that extension in the offer.

Once an agent has sent its offer or its answer, that agent **MUST** be prepared to receive both STUN and media packets on each candidate. As discussed in Section 10.1, media packets can be sent to a candidate prior to its appearance as the default destination for media in an offer or answer.

5. Receiving the Initial Offer

When an agent receives an initial offer, it will check if the offerer supports ICE, determine its own role, gather candidates, prioritize them, choose default candidates, encode and send an answer, and for full implementations, form the check lists and begin connectivity checks.

5.1. Verifying ICE Support

Certain middleboxes, such as ALGs, may alter the ICE offer and/or answer in a way that breaks ICE. If the using protocol is vulnerable to this kind of changes, called ICE mismatch, the answerer needs to detect this and signal this back to the offerer. The details on

whether this is needed and how it is done is defined by the usage specifications.

5.2. Determining Role

For each session, each agent takes on a role. There are two roles -- controlling and controlled. The controlling agent is responsible for the choice of the final candidate pairs used for communications. For a full agent, this means nominating the candidate pairs that can be used by ICE for each media stream, and for generating the updated offer based on ICE's selection, when needed. For a lite implementation, being the controlling agent means selecting a candidate pair based on the ones in the offer and answer (for IPv4, there is only ever one pair), and then generating an updated offer reflecting that selection, when needed (it is never needed for an IPv4-only host). The controlled agent is told which candidate pairs to use for each media stream, and does not generate an updated offer to signal this information. The sections below describe in detail the actual procedures followed by controlling and controlled nodes.

The rules for determining the role and the impact on behavior are as follows:

Both agents are full: The agent that generated the offer which started the ICE processing MUST take the controlling role, and the other MUST take the controlled role. Both agents will form check lists, run the ICE state machines, and generate connectivity checks. The controlling agent will execute the logic in Section 8.1 to nominate pairs that will be selected by ICE, and then both agents end ICE as described in Section 8.1.2.

One agent full, one lite: The full agent MUST take the controlling role, and the lite agent MUST take the controlled role. The full agent will form check lists, run the ICE state machines, and generate connectivity checks. That agent will execute the logic in Section 8.1 to nominate pairs that will be selected by ICE, and use the logic in Section 8.1.2 to end ICE. The lite implementation will just listen for connectivity checks, receive them and respond to them, and then conclude ICE as described in Section 8.2. For the lite implementation, the state of ICE processing for each media stream is considered to be Running, and the state of ICE overall is Running.

Both lite: The agent that generated the offer which started the ICE processing MUST take the controlling role, and the other MUST take the controlled role. In this case, no connectivity checks are ever sent. Rather, once the offer/answer exchange completes, each agent performs the processing described in Section 8 without

connectivity checks. It is possible that both agents will believe they are controlled or controlling. In the latter case, the conflict is resolved through glare detection capabilities in the signaling protocol carrying the offer/answer exchange. The state of ICE processing for each media stream is considered to be Running, and the state of ICE overall is Running.

Once roles are determined for a session, they persist unless ICE is restarted. An ICE restart causes a new selection of roles and tie-breakers.

5.3. Gathering Candidates

The process for gathering candidates at the answerer is identical to the process for the offerer as described in Section 4.1.1 for full implementations and Section 4.2 for lite implementations. It is RECOMMENDED that this process begin immediately on receipt of the offer, prior to alerting the user. Such gathering MAY begin when an agent starts.

5.4. Prioritizing Candidates

The process for prioritizing candidates at the answerer is identical to the process followed by the offerer, as described in Section 4.1.2 for full implementations and Section 4.2 for lite implementations.

5.5. Encoding the Answer

The process for encoding the answer is identical to the process followed by the offerer for both full and lite implementations, as described in Section 4.3.

5.6. Forming the Check Lists

Forming check lists is done only by full implementations. Lite implementations MUST skip the steps defined in this section.

There is one check list per in-use media stream resulting from the offer/answer exchange. To form the check list for a media stream, the agent forms candidate pairs, computes a candidate pair priority, orders the pairs by priority, prunes them, and sets their states. These steps are described in this section.

5.6.1. Forming Candidate Pairs

First, the agent takes each of its candidates for a media stream (called LOCAL CANDIDATES) and pairs them with the candidates it received from its peer (called REMOTE CANDIDATES) for that media

stream. In order to prevent the attacks described in Section 14.4.1, agents MAY limit the number of candidates they'll accept in an offer or answer. A local candidate is paired with a remote candidate if and only if the two candidates have the same component ID and have the same IP address version. It is possible that some of the local candidates won't get paired with remote candidates, and some of the remote candidates won't get paired with local candidates. This can happen if one agent doesn't include candidates for the all of the components for a media stream. If this happens, the number of components for that media stream is effectively reduced, and considered to be equal to the minimum across both agents of the maximum component ID provided by each agent across all components for the media stream.

In the case of RTP, this would happen when one agent provides candidates for RTCP, and the other does not. As another example, the offerer can multiplex RTP and RTCP on the same port and signals that it can do that in the SDP through an SDP attribute [RFC5761]. However, since the offerer doesn't know if the answerer can perform such multiplexing, the offerer includes candidates for RTP and RTCP on separate ports, so that the offer has two components per media stream. If the answerer can perform such multiplexing, it would include just a single component for each candidate -- for the combined RTP/RTCP mux. ICE would end up acting as if there was just a single component for this candidate.

With IPv6 it is common for a host to have multiple host candidates for each interface. To keep the amount of resulting candidate pairs reasonable and to avoid candidate pairs that are highly unlikely to work, IPv6 link-local addresses [RFC4291] MUST NOT be paired with other than link-local addresses.

The candidate pairs whose local and remote candidates are both the default candidates for a particular component is called, unsurprisingly, the default candidate pair for that component. This is the pair that would be used to transmit media if both agents had not been ICE aware.

In order to aid understanding, Figure 6 shows the relationships between several key concepts -- transport addresses, candidates, candidate pairs, and check lists, in addition to indicating the main properties of candidates and candidate pairs.

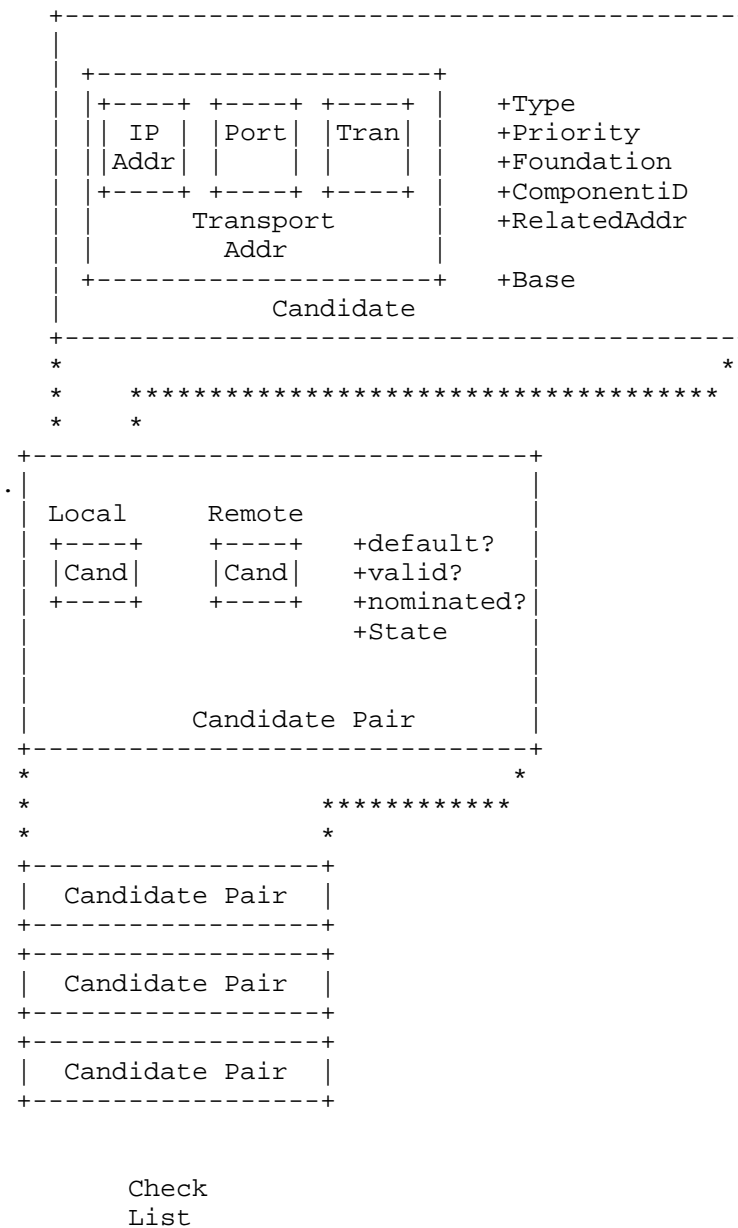


Figure 6: Conceptual Diagram of a Check List

5.6.2. Computing Pair Priority and Ordering Pairs

Once the pairs are formed, a candidate pair priority is computed. Let G be the priority for the candidate provided by the controlling agent. Let D be the priority for the candidate provided by the controlled agent. The priority for a pair is computed as:

$$\text{pair priority} = 2^{32} * \text{MIN}(G, D) + 2 * \text{MAX}(G, D) + (G > D ? 1 : 0)$$

Where $G > D ? 1 : 0$ is an expression whose value is 1 if G is greater than D , and 0 otherwise. Once the priority is assigned, the agent sorts the candidate pairs in decreasing order of priority. If two pairs have identical priority, the ordering amongst them is arbitrary.

5.6.3. Pruning the Pairs

This sorted list of candidate pairs is used to determine a sequence of connectivity checks that will be performed. Each check involves sending a request from a local candidate to a remote candidate. Since an agent cannot send requests directly from a reflexive candidate, but only from its base, the agent next goes through the sorted list of candidate pairs. For each pair where the local candidate is server reflexive, the server reflexive candidate MUST be replaced by its base. Once this has been done, the agent MUST prune the list. This is done by removing a pair if its local and remote candidates are identical to the local and remote candidates of a pair higher up on the priority list. The result is a sequence of ordered candidate pairs, called the check list for that media stream.

In addition, in order to limit the attacks described in Section 14.4.1, an agent MUST limit the total number of connectivity checks the agent performs across all check lists to a specific value, and this value MUST be configurable. A default of 100 is RECOMMENDED. This limit is enforced by discarding the lower-priority candidate pairs until there are less than 100. It is RECOMMENDED that a lower value be utilized when possible, set to the maximum number of plausible checks that might be seen in an actual deployment configuration. The requirement for configuration is meant to provide a tool for fixing this value in the field if, once deployed, it is found to be problematic.

5.6.4. Computing States

Each candidate pair in the check list has a foundation and a state. The foundation is the combination of the foundations of the local and remote candidates in the pair. The state is assigned once the check list for each media stream has been computed. There are five potential values that the state can have:

Waiting: A check has not been performed for this pair, and can be performed as soon as it is the highest-priority Waiting pair on the check list.

In-Progress: A check has been sent for this pair, but the transaction is in progress.

Succeeded: A check for this pair was already done and produced a successful result.

Failed: A check for this pair was already done and failed, either never producing any response or producing an unrecoverable failure response.

Frozen: A check for this pair hasn't been performed, and it can't yet be performed until some other check succeeds, allowing this pair to unfreeze and move into the Waiting state.

As ICE runs, the pairs will move between states as shown in Figure 7.

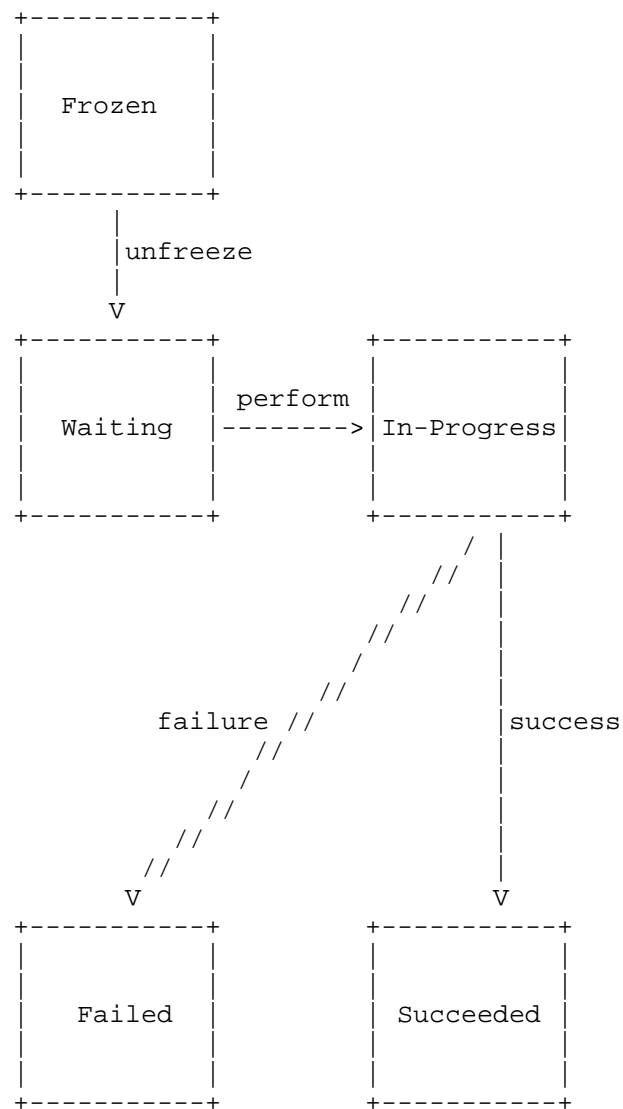


Figure 7: Pair State FSM

The initial states for each pair in a check list are computed by performing the following sequence of steps:

1. The agent sets all of the pairs in each check list to the Frozen state.

2. The agent examines the check list for the first media stream.
For that media stream:
 - * For all pairs with the same foundation, it sets the state of the pair with the lowest component ID to Waiting. If there is more than one such pair, the one with the highest-priority is used.

One of the check lists will have some number of pairs in the Waiting state, and the other check lists will have all of their pairs in the Frozen state. A check list with at least one pair that is Waiting is called an active check list, and a check list with all pairs Frozen is called a frozen check list.

The check list itself is associated with a state, which captures the state of ICE checks for that media stream. There are three states:

Running: In this state, ICE checks are still in progress for this media stream.

Completed: In this state, ICE checks have produced nominated pairs for each component of the media stream. Consequently, ICE has succeeded and media can be sent.

Failed: In this state, the ICE checks have not completed successfully for this media stream.

When a check list is first constructed as the consequence of an offer/answer exchange, it is placed in the Running state.

ICE processing across all media streams also has a state associated with it. This state is equal to Running while ICE processing is under way. The state is Completed when ICE processing is complete and Failed if it failed without success. Rules for transitioning between states are described below.

5.7. Scheduling Checks

Checks are generated only by full implementations. Lite implementations MUST skip the steps described in this section.

An agent performs ordinary checks and triggered checks. The generation of both checks is governed by a timer that fires periodically for each media stream. The agent maintains a FIFO queue, called the triggered check queue, which contains candidate pairs for which checks are to be sent at the next available opportunity. When the timer fires, the agent removes the top pair from the triggered check queue, performs a connectivity check on that

pair, and sets the state of the candidate pair to In-Progress. If there are no pairs in the triggered check queue, an ordinary check is sent.

Once the agent has computed the check lists as described in Section 5.6, it sets a timer for each active check list. The timer fires every $T_a \cdot N$ seconds, where N is the number of active check lists (initially, there is only one active check list). Implementations MAY set the timer to fire less frequently than this. Implementations SHOULD take care to spread out these timers so that they do not fire at the same time for each media stream. T_a and the retransmit timer RTO are computed as described in Section 12. Multiplying by N allows this aggregate check throughput to be split between all active check lists. The first timer fires immediately, so that the agent performs a connectivity check the moment the offer/answer exchange has been done, followed by the next check T_a seconds later (since there is only one active check list).

When the timer fires and there is no triggered check to be sent, the agent MUST choose an ordinary check as follows:

- o Find the highest-priority pair in that check list that is in the Waiting state.
- o If there is such a pair:
 - * Send a STUN check from the local candidate of that pair to the remote candidate of that pair. The procedures for forming the STUN request for this purpose are described in Section 7.1.2.
 - * Set the state of the candidate pair to In-Progress.
- o If there is no such pair:
 - * Find the highest-priority pair in that check list that is in the Frozen state.
 - * If there is such a pair:
 - + Unfreeze the pair.
 - + Perform a check for that pair, causing its state to transition to In-Progress.
 - * If there is no such pair:
 - + Terminate the timer for that check list.

To compute the message integrity for the check, the agent uses the remote username fragment and password learned from the offer or answer from its peer. The local username fragment is known directly by the agent for its own candidate.

6. Receipt of the Initial Answer

This section describes the procedures that an agent follows when it receives the answer from the peer. It verifies that its peer supports ICE, determines its role, and for full implementations, forms the check list and begins performing ordinary checks.

6.1. Verifying ICE Support

The logic at the offerer is identical to that of the answerer as described in Section 5.1, with the exception that an offerer would not ever indicate ICE mismatch.

6.2. Determining Role

The offerer follows the same procedures described for the answerer in Section 5.2.

6.3. Forming the Check List

Formation of check lists is performed only by full implementations. The offerer follows the same procedures described for the answerer in Section 5.6.

6.4. Performing Ordinary Checks

Ordinary checks are performed only by full implementations. The offerer follows the same procedures described for the answerer in Section 5.7.

7. Performing Connectivity Checks

This section describes how connectivity checks are performed. All ICE implementations are required to be compliant to [RFC5389], as opposed to the older [RFC3489]. However, whereas a full implementation will both generate checks (acting as a STUN client) and receive them (acting as a STUN server), a lite implementation will only receive checks, and thus will only act as a STUN server.

7.1. STUN Client Procedures

These procedures define how an agent sends a connectivity check, whether it is an ordinary or a triggered check. These procedures are only applicable to full implementations.

7.1.1. Creating Permissions for Relayed Candidates

If the connectivity check is being sent using a relayed local candidate, the client **MUST** create a permission first if it has not already created one previously. It would have created one previously if it had told the TURN server to create a permission for the given relayed candidate towards the IP address of the remote candidate. To create the permission, the agent follows the procedures defined in [RFC5766]. The permission **MUST** be created towards the IP address of the remote candidate. It is **RECOMMENDED** that the agent defer creation of a TURN channel until ICE completes, in which case permissions for connectivity checks are normally created using a CreatePermission request. Once established, the agent **MUST** keep the permission active until ICE concludes.

7.1.2. Sending the Request

A connectivity check is generated by sending a Binding request from a local candidate to a remote candidate. [RFC5389] describes how Binding requests are constructed and generated. A connectivity check **MUST** utilize the STUN short-term credential mechanism. Support for backwards compatibility with RFC 3489 **MUST NOT** be used or assumed with connectivity checks. The FINGERPRINT mechanism **MUST** be used for connectivity checks.

ICE extends STUN by defining several new attributes, including PRIORITY, USE-CANDIDATE, ICE-CONTROLLED, and ICE-CONTROLLING. These new attributes are formally defined in Section 15.1, and their usage is described in the subsections below. These STUN extensions are applicable only to connectivity checks used for ICE.

7.1.2.1. PRIORITY and USE-CANDIDATE

An agent **MUST** include the PRIORITY attribute in its Binding request. The attribute **MUST** be set equal to the priority that would be assigned, based on the algorithm in Section 4.1.2, to a peer reflexive candidate, should one be learned as a consequence of this check (see Section 7.1.3.2.1 for how peer reflexive candidates are learned). This priority value will be computed identically to how the priority for the local candidate of the pair was computed, except that the type preference is set to the value for peer reflexive candidate types.

The controlling agent MAY include the USE-CANDIDATE attribute in the Binding request. The controlled agent MUST NOT include it in its Binding request. This attribute signals that the controlling agent wishes to cease checks for this component, and use the candidate pair resulting from the check for this component. Section 8.1.1 provides guidance on determining when to include it.

7.1.2.2. ICE-CONTROLLED and ICE-CONTROLLING

The agent MUST include the ICE-CONTROLLED attribute in the request if it is in the controlled role, and MUST include the ICE-CONTROLLING attribute in the request if it is in the controlling role. The content of either attribute MUST be the tie-breaker that was determined in Section 5.2. These attributes are defined fully in Section 15.1.

7.1.2.3. Forming Credentials

A Binding request serving as a connectivity check MUST utilize the STUN short-term credential mechanism. The username for the credential is formed by concatenating the username fragment provided by the peer with the username fragment of the agent sending the request, separated by a colon (":"). The password is equal to the password provided by the peer. For example, consider the case where agent L is the offerer, and agent R is the answerer. Agent L included a username fragment of LFRAG for its candidates and a password of LPASS. Agent R provided a username fragment of RFRAG and a password of RPASS. A connectivity check from L to R utilizes the username RFRAG:LFRAG and a password of RPASS. A connectivity check from R to L utilizes the username LFRAG:RFRAG and a password of LPASS. The responses utilize the same usernames and passwords as the requests (note that the USERNAME attribute is not present in the response).

7.1.2.4. DiffServ Treatment

If the agent is using Diffserv Codepoint markings [RFC2475] in its media packets, it SHOULD apply those same markings to its connectivity checks.

7.1.3. Processing the Response

When a Binding response is received, it is correlated to its Binding request using the transaction ID, as defined in [RFC5389], which then ties it to the candidate pair for which the Binding request was sent. This section defines additional procedures for processing Binding responses specific to this usage of STUN.

7.1.3.1. Failure Cases

If the STUN transaction generates a 487 (Role Conflict) error response, the agent checks whether it included the ICE-CONTROLLED or ICE-CONTROLLING attribute in the Binding request. If the request contained the ICE-CONTROLLED attribute, the agent MUST switch to the controlling role if it has not already done so. If the request contained the ICE-CONTROLLING attribute, the agent MUST switch to the controlled role if it has not already done so. Once it has switched, the agent MUST enqueue the candidate pair whose check generated the 487 into the triggered check queue. The state of that pair is set to Waiting. When the triggered check is sent, it will contain an ICE-CONTROLLING or ICE-CONTROLLED attribute reflecting its new role. Note, however, that the tie-breaker value MUST NOT be reselected.

A change in roles will require an agent to recompute pair priorities (Section 5.6.2), since those priorities are a function of controlling and controlled roles. The change in role will also impact whether the agent is responsible for selecting nominated pairs and generating updated offers upon conclusion of ICE.

Agents MAY support receipt of ICMP errors for connectivity checks. If the STUN transaction generates an ICMP error, the agent sets the state of the pair to Failed. If the STUN transaction generates a STUN error response that is unrecoverable (as defined in [RFC5389]) or times out, the agent sets the state of the pair to Failed.

The agent MUST check that the source IP address and port of the response equal the destination IP address and port to which the Binding request was sent, and that the destination IP address and port of the response match the source IP address and port from which the Binding request was sent. In other words, the source and destination transport addresses in the request and responses are symmetric. If they are not symmetric, the agent sets the state of the pair to Failed.

7.1.3.2. Success Cases

A check is considered to be a success if all of the following are true:

- o The STUN transaction generated a success response.
- o The source IP address and port of the response equals the destination IP address and port to which the Binding request was sent.

- o The destination IP address and port of the response match the source IP address and port from which the Binding request was sent.

7.1.3.2.1. Discovering Peer Reflexive Candidates

The agent checks the mapped address from the STUN response. If the transport address does not match any of the local candidates that the agent knows about, the mapped address represents a new candidate -- a peer reflexive candidate. Like other candidates, it has a type, base, priority, and foundation. They are computed as follows:

- o Its type is equal to peer reflexive.
- o Its base is set equal to the local candidate of the candidate pair from which the STUN check was sent.
- o Its priority is set equal to the value of the PRIORITY attribute in the Binding request.
- o Its foundation is selected as described in Section 4.1.1.3.

This peer reflexive candidate is then added to the list of local candidates for the media stream. Its username fragment and password are the same as all other local candidates for that media stream. However, the peer reflexive candidate is not paired with other remote candidates. This is not necessary; a valid pair will be generated from it momentarily based on the procedures in Section 7.1.3.2.2. If an agent wishes to pair the peer reflexive candidate with other remote candidates besides the one in the valid pair that will be generated, the agent MAY generate an updated offer which includes the peer reflexive candidate. This will cause it to be paired with all other remote candidates.

7.1.3.2.2. Constructing a Valid Pair

The agent constructs a candidate pair whose local candidate equals the mapped address of the response, and whose remote candidate equals the destination address to which the request was sent. This is called a valid pair, since it has been validated by a STUN connectivity check. The valid pair may equal the pair that generated the check, may equal a different pair in the check list, or may be a pair not currently on any check list. If the pair equals the pair that generated the check or is on a check list currently, it is also added to the VALID LIST, which is maintained by the agent for each media stream. This list is empty at the start of ICE processing, and fills as checks are performed, resulting in valid candidate pairs.

It will be very common that the pair will not be on any check list. Recall that the check list has pairs whose local candidates are never server reflexive; those pairs had their local candidates converted to the base of the server reflexive candidates, and then pruned if they were redundant. When the response to the STUN check arrives, the mapped address will be reflexive if there is a NAT between the two. In that case, the valid pair will have a local candidate that doesn't match any of the pairs in the check list.

If the pair is not on any check list, the agent computes the priority for the pair based on the priority of each candidate, using the algorithm in Section 5.6. The priority of the local candidate depends on its type. If it is not peer reflexive, it is equal to the priority signaled for that candidate in the offer or answer. If it is peer reflexive, it is equal to the PRIORITY attribute the agent placed in the Binding request that just completed. The priority of the remote candidate is taken from the offer/answer of the peer. If the candidate does not appear there, then the check must have been a triggered check to a new remote candidate. In that case, the priority is taken as the value of the PRIORITY attribute in the Binding request that triggered the check that just completed. The pair is then added to the VALID LIST.

7.1.3.2.3. Updating Pair States

The agent sets the state of the pair that *generated* the check to Succeeded. Note that, the pair which *generated* the check may be different than the valid pair constructed in Section 7.1.3.2.2 as a consequence of the response. The success of this check might also cause the state of other checks to change as well. The agent MUST perform the following two steps:

1. The agent changes the states for all other Frozen pairs for the same media stream and same foundation to Waiting. Typically, but not always, these other pairs will have different component IDs.
2. If there is a pair in the valid list for every component of this media stream (where this is the actual number of components being used, in cases where the number of components signaled in the offer/answer differs from offerer to answerer), the success of this check may unfreeze checks for other media streams. Note that this step is followed not just the first time the valid list under consideration has a pair for every component, but every subsequent time a check succeeds and adds yet another pair to that valid list. The agent examines the check list for each other media stream in turn:

- * If the check list is active, the agent changes the state of all Frozen pairs in that check list whose foundation matches a pair in the valid list under consideration to Waiting.
- * If the check list is frozen, and there is at least one pair in the check list whose foundation matches a pair in the valid list under consideration, the state of all pairs in the check list whose foundation matches a pair in the valid list under consideration is set to Waiting. This will cause the check list to become active, and ordinary checks will begin for it, as described in Section 5.7.
- * If the check list is frozen, and there are no pairs in the check list whose foundation matches a pair in the valid list under consideration, the agent
 - + groups together all of the pairs with the same foundation, and
 - + for each group, sets the state of the pair with the lowest component ID to Waiting. If there is more than one such pair, the one with the highest-priority is used.

7.1.3.2.4. Updating the Nominated Flag

If the agent was a controlling agent, and it had included a USE-CANDIDATE attribute in the Binding request, the valid pair generated from that check has its nominated flag set to true. This flag indicates that this valid pair should be used for media if it is the highest-priority one amongst those whose nominated flag is set. This may conclude ICE processing for this media stream or all media streams; see Section 8.

If the agent is the controlled agent, the response may be the result of a triggered check that was sent in response to a request that itself had the USE-CANDIDATE attribute. This case is described in Section 7.2.1.5, and may now result in setting the nominated flag for the pair learned from the original request.

7.1.3.3. Check List and Timer State Updates

Regardless of whether the check was successful or failed, the completion of the transaction may require updating of check list and timer states.

If all of the pairs in the check list are now either in the Failed or Succeeded state:

- o 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.
- o For each frozen check list, the agent
 - * groups together all of the pairs with the same foundation, and
 - * for each group, sets the state of the pair with the lowest component ID to Waiting. If there is more than one such pair, the one with the highest-priority is used.

If none of the pairs in the check list are in the Waiting or Frozen state, the check list is no longer considered active, and will not count towards the value of N in the computation of timers for ordinary checks as described in Section 5.7.

7.2. STUN Server Procedures

An agent **MUST** be prepared to receive a Binding request on the base of each candidate it included in its most recent offer or answer. This requirement holds even if the peer is a lite implementation.

The agent **MUST** use the short-term credential mechanism (i.e., the MESSAGE-INTEGRITY attribute) to authenticate the request and perform a message integrity check. Likewise, the short-term credential mechanism **MUST** be used for the response. The agent **MUST** consider the username to be valid if it consists of two values separated by a colon, where the first value is equal to the username fragment generated by the agent in an offer or answer for a session in-progress. It is possible (and in fact very likely) that an offerer will receive a Binding request prior to receiving the answer from its peer. If this happens, the agent **MUST** immediately generate a response (including computation of the mapped address as described in Section 7.2.1.2). The agent has sufficient information at this point to generate the response; the password from the peer is not required. Once the answer is received, it **MUST** proceed with the remaining steps required, namely, Section 7.2.1.3, Section 7.2.1.4, and Section 7.2.1.5 for full implementations. In cases where multiple STUN requests are received before the answer, this may cause several pairs to be queued up in the triggered check queue.

An agent **MUST NOT** utilize the ALTERNATE-SERVER mechanism, and **MUST NOT** support the backwards-compatibility mechanisms to RFC 3489. It **MUST** utilize the FINGERPRINT mechanism.

If the agent is using Diffserv Codepoint markings [RFC2475] in its media packets, it **SHOULD** apply those same markings to its responses to Binding requests. The same would apply to any layer 2 markings

the endpoint might be applying to media packets.

7.2.1. Additional Procedures for Full Implementations

This subsection defines the additional server procedures applicable to full implementations.

7.2.1.1. Detecting and Repairing Role Conflicts

Normally, the rules for selection of a role in Section 5.2 will result in each agent selecting a different role -- one controlling and one controlled. However, in unusual call flows, typically utilizing third party call control, it is possible for both agents to select the same role. This section describes procedures for checking for this case and repairing it. These procedures apply only to usages of ICE that require conflict resolution. The usage document MUST specify whether this mechanism is needed.

An agent MUST examine the Binding request for either the ICE-CONTROLLING or ICE-CONTROLLED attribute. It MUST follow these procedures:

- o If neither ICE-CONTROLLING nor ICE-CONTROLLED is present in the request, the peer agent may have implemented a previous version of this specification. There may be a conflict, but it cannot be detected.
- o If the agent is in the controlling role, and the ICE-CONTROLLING attribute is present in the request:
 - * If the agent's tie-breaker is larger than or equal to the contents of the ICE-CONTROLLING attribute, the agent generates a Binding error response and includes an ERROR-CODE attribute with a value of 487 (Role Conflict) but retains its role.
 - * If the agent's tie-breaker is less than the contents of the ICE-CONTROLLING attribute, the agent switches to the controlled role.
- o If the agent is in the controlled role, and the ICE-CONTROLLED attribute is present in the request:
 - * If the agent's tie-breaker is larger than or equal to the contents of the ICE-CONTROLLED attribute, the agent switches to the controlling role.
 - * If the agent's tie-breaker is less than the contents of the ICE-CONTROLLED attribute, the agent generates a Binding error

response and includes an ERROR-CODE attribute with a value of 487 (Role Conflict) but retains its role.

- o If the agent is in the controlled role and the ICE-CONTROLLING attribute was present in the request, or the agent was in the controlling role and the ICE-CONTROLLED attribute was present in the request, there is no conflict.

A change in roles will require an agent to recompute pair priorities (Section 5.6.2), since those priorities are a function of controlling and controlled roles. The change in role will also impact whether the agent is responsible for selecting nominated pairs and generated updated offers upon conclusion of ICE.

The remaining sections in Section 7.2.1 are followed if the server generated a successful response to the Binding request, even if the agent changed roles.

7.2.1.2. Computing Mapped Address

For requests being received on a relayed candidate, the source transport address used for STUN processing (namely, generation of the XOR-MAPPED-ADDRESS attribute) is the transport address as seen by the TURN server. That source transport address will be present in the XOR-PEER-ADDRESS attribute of a Data Indication message, if the Binding request was delivered through a Data Indication. If the Binding request was delivered through a ChannelData message, the source transport address is the one that was bound to the channel.

7.2.1.3. Learning Peer Reflexive Candidates

If the source transport address of the request does not match any existing remote candidates, it represents a new peer reflexive remote candidate. This candidate is constructed as follows:

- o The priority of the candidate is set to the PRIORITY attribute from the request.
- o The type of the candidate is set to peer reflexive.
- o The foundation of the candidate is set to an arbitrary value, different from the foundation for all other remote candidates. If any subsequent offer/answer exchanges contain this peer reflexive candidate, it will signal the actual foundation for the candidate.
- o The component ID of this candidate is set to the component ID for the local candidate to which the request was sent.

This candidate is added to the list of remote candidates. However, the agent does not pair this candidate with any local candidates.

7.2.1.4. Triggered Checks

Next, the agent constructs a pair whose local candidate is equal to the transport address on which the STUN request was received, and a remote candidate equal to the source transport address where the request came from (which may be the peer reflexive remote candidate that was just learned). The local candidate will either be a host candidate (for cases where the request was not received through a relay) or a relayed candidate (for cases where it is received through a relay). The local candidate can never be a server reflexive candidate. Since both candidates are known to the agent, it can obtain their priorities and compute the candidate pair priority. This pair is then looked up in the check list. There can be one of several outcomes:

- o If the pair is already on the check list:
 - * If the state of that pair is Waiting or Frozen, a check for that pair is enqueued into the triggered check queue if not already present.
 - * If the state of that pair is In-Progress, the agent cancels the in-progress transaction. Cancellation means that the agent will not retransmit the request, will not treat the lack of response to be a failure, but will wait the duration of the transaction timeout for a response. In addition, the agent MUST create a new connectivity check for that pair (representing a new STUN Binding request transaction) by enqueueing the pair in the triggered check queue. The state of the pair is then changed to Waiting.
 - * If the state of the pair is Failed, it is changed to Waiting and the agent MUST create a new connectivity check for that pair (representing a new STUN Binding request transaction), by enqueueing the pair in the triggered check queue.
 - * If the state of that pair is Succeeded, nothing further is done.

These steps are done to facilitate rapid completion of ICE when both agents are behind NAT.

- o If the pair is not already on the check list:

- * The pair is inserted into the check list based on its priority.
- * Its state is set to Waiting.
- * The pair is enqueued into the triggered check queue.

When a triggered check is to be sent, it is constructed and processed as described in Section 7.1.2. These procedures require the agent to know the transport address, username fragment, and password for the peer. The username fragment for the remote candidate is equal to the part after the colon of the USERNAME in the Binding request that was just received. Using that username fragment, the agent can check the offers/answers received from its peer (there may be more than one in cases of forking), and find this username fragment. The corresponding password is then selected.

7.2.1.5. Updating the Nominated Flag

If the Binding request received by the agent had the USE-CANDIDATE attribute set, and the agent is in the controlled role, the agent looks at the state of the pair computed in Section 7.2.1.4:

- o If the state of this pair is Succeeded, it means that the check generated by this pair produced a successful response. This would have caused the agent to construct a valid pair when that success response was received (see Section 7.1.3.2.2). The agent now sets the nominated flag in the valid pair to true. This may end ICE processing for this media stream; see Section 8.
- o If the state of this pair is In-Progress, if its check produces a successful result, the resulting valid pair has its nominated flag set when the response arrives. This may end ICE processing for this media stream when it arrives; see Section 8.

7.2.2. Additional Procedures for Lite Implementations

If the check that was just received contained a USE-CANDIDATE attribute, the agent constructs a candidate pair whose local candidate is equal to the transport address on which the request was received, and whose remote candidate is equal to the source transport address of the request that was received. This candidate pair is assigned an arbitrary priority, and placed into a list of valid candidates called the valid list. The agent sets the nominated flag for that pair to true. ICE processing is considered complete for a media stream if the valid list contains a candidate pair for each component.

8. Concluding ICE Processing

This section describes how an agent completes ICE.

8.1. Procedures for Full Implementations

Concluding ICE involves nominating pairs by the controlling agent and updating of state machinery.

8.1.1. Nominating Pairs

The controlling agent nominates pairs to be selected by ICE by using one of two techniques: regular nomination or aggressive nomination. If its peer has a lite implementation, an agent **MUST** use a regular nomination algorithm. If its peer is using ICE options (present in an ice-options attribute from the peer) that the agent does not understand, the agent **MUST** use a regular nomination algorithm. If its peer is a full implementation and isn't using any ICE options or is using ICE options understood by the agent, the agent **MAY** use either the aggressive or the regular nomination algorithm. However, the regular algorithm is **RECOMMENDED** since it provides greater stability.

8.1.1.1. Regular Nomination

With regular nomination, the agent lets some number of checks complete, each of which omit the USE-CANDIDATE attribute. Once one or more checks complete successfully for a component of a media stream, valid pairs are generated and added to the valid list. The agent lets the checks continue until some stopping criterion is met, and then picks amongst the valid pairs based on an evaluation criterion. The criteria for stopping the checks and for evaluating the valid pairs is entirely a matter of local optimization.

When the controlling agent selects the valid pair, it repeats the check that produced this valid pair (by enqueueing the pair that generated the check into the triggered check queue), this time with the USE-CANDIDATE attribute. This check should succeed (since the previous did), causing the nominated flag of that and only that pair to be set. Consequently, there will be only a single nominated pair in the valid list for each component, and when the state of the check list moves to completed, that exact pair is selected by ICE for sending and receiving media for that component.

Regular nomination provides the most flexibility, since the agent has control over the stopping and selection criteria for checks. The only requirement is that the agent **MUST** eventually pick one and only one candidate pair and generate a check for that pair with the USE-

CANDIDATE attribute present. Regular nomination also improves ICE's resilience to variations in implementation (see Section 11). Regular nomination is also more stable, allowing both agents to converge on a single pair for media without any transient selections, which can happen with the aggressive algorithm. The drawback of regular nomination is that it is guaranteed to increase latencies because it requires an additional check to be done.

8.1.1.2. Aggressive Nomination

With aggressive nomination, the controlling agent includes the USE-CANDIDATE attribute in every check it sends. Once the first check for a component succeeds, it will be added to the valid list and have its nominated flag set. When all components have a nominated pair in the valid list, media can begin to flow using the highest-priority nominated pair. However, because the agent included the USE-CANDIDATE attribute in all of its checks, another check may yet complete, causing another valid pair to have its nominated flag set. ICE always selects the highest-priority nominated candidate pair from the valid list as the one used for media. Consequently, the selected pair may actually change briefly as ICE checks complete, resulting in a set of transient selections until it stabilizes.

8.1.2. Updating States

For both controlling and controlled agents, the state of ICE processing depends on the presence of nominated candidate pairs in the valid list and on the state of the check list. Note that, at any time, more than one of the following cases can apply:

- o If there are no nominated pairs in the valid list for a media stream and the state of the check list is Running, ICE processing continues.
- o If there is at least one nominated pair in the valid list for a media stream and the state of the check list is Running:
 - * The agent MUST remove all Waiting and Frozen pairs in the check list and triggered check queue for the same component as the nominated pairs for that media stream.
 - * If an In-Progress pair in the check list is for the same component as a nominated pair, the agent SHOULD cease retransmissions for its check if its pair priority is lower than the lowest-priority nominated pair for that component.

- o Once there is at least one nominated pair in the valid list for every component of at least one media stream and the state of the check list is Running:
 - * The agent MUST change the state of processing for its check list for that media stream to Completed.
 - * The agent MUST continue to respond to any checks it may still receive for that media stream, and MUST perform triggered checks if required by the processing of Section 7.2.
 - * The agent MUST continue retransmitting any In-Progress checks for that check list.
 - * The agent MAY begin transmitting media for this media stream as described in Section 10.1.
- o Once the state of each check list is Completed:
 - * The agent sets the state of ICE processing overall to Completed.
 - * If the controlling agent is using an aggressive nomination algorithm, this may result in several updated offers as the pairs selected for media change. An agent MAY delay sending the offer for a brief interval (one second is RECOMMENDED) in order to allow the selected pairs to stabilize.
- o If the state of the check list is Failed, ICE has not been able to complete for this media stream. The correct behavior depends on the state of the check lists for other media streams:
 - * If all check lists are Failed, ICE processing overall is considered to be in the Failed state, and the agent SHOULD consider the session a failure, SHOULD NOT restart ICE, and the controlling agent SHOULD terminate the entire session.
 - * If at least one of the check lists for other media streams is Completed, the controlling agent SHOULD remove the failed media stream from the session in its updated offer.
 - * If none of the check lists for other media streams are Completed, but at least one is Running, the agent SHOULD let ICE continue.

8.2. Procedures for Lite Implementations

Concluding ICE for a lite implementation is relatively straightforward. There are two cases to consider:

The implementation is lite, and its peer is full.

The implementation is lite, and its peer is lite.

The effect of ICE concluding is that the agent can free any allocated host candidates that were not utilized by ICE, as described in Section 8.3.

8.2.1. Peer Is Full

In this case, the agent will receive connectivity checks from its peer. When an agent has received a connectivity check that includes the USE-CANDIDATE attribute for each component of a media stream, the state of ICE processing for that media stream moves from Running to Completed. When the state of ICE processing for all media streams is Completed, the state of ICE processing overall is Completed.

The lite implementation will never itself determine that ICE processing has failed for a media stream; rather, the full peer will make that determination and then remove or restart the failed media stream in a subsequent offer.

8.2.2. Peer Is Lite

Once the offer/answer exchange has completed, both agents examine their candidates and those of its peer. For each media stream, each agent pairs up its own candidates with the candidates of its peer for that media stream. Two candidates are paired up when they are for the same component, utilize the same transport protocol (UDP in this specification), and are from the same IP address family (IPv4 or IPv6).

- o If there is a single pair per component, that pair is added to the Valid list. If all of the components for a media stream had one pair, the state of ICE processing for that media stream is set to Completed. If all media streams are Completed, the state of ICE processing is set to Completed overall. This will always be the case for implementations that are IPv4-only.

- o If there is more than one pair per component:

- * The agent MUST select a pair based on local policy. Since this case only arises for IPv6, it is RECOMMENDED that an agent

follow the procedures of RFC 6724 [RFC6724] to select a single pair.

- * The agent adds the selected pair for each component to the valid list. As described in Section 10.1, this will permit media to begin flowing. However, it is possible (and in fact likely) that both agents have chosen different pairs.
- * To reconcile this, the controlling agent **MUST** send an updated offer which will include the remote-candidates attribute.
- * The agent **MUST NOT** update the state of ICE processing when the offer is sent. If this subsequent offer completes, the controlling agent **MUST** change the state of ICE processing to Completed for all media streams, and the state of ICE processing overall to Completed.

8.3. Freeing Candidates

8.3.1. Full Implementation Procedures

The procedures in Section 8 require that an agent continue to listen for STUN requests and continue to generate triggered checks for a media stream, even once processing for that stream completes. The rules in this section describe when it is safe for an agent to cease sending or receiving checks on a candidate that was not selected by ICE, and then free the candidate.

8.3.2. Lite Implementation Procedures

A lite implementation **MAY** free candidates not selected by ICE as soon as ICE processing has reached the Completed state for all peers for all media streams using those candidates.

9. Keepalives

All endpoints **MUST** send keepalives for each media session. These keepalives serve the purpose of keeping NAT bindings alive for the media session. These keepalives **MUST** be sent even if ICE is not being utilized for the session at all. The keepalive **SHOULD** be sent using a format that is supported by its peer. ICE endpoints allow for STUN-based keepalives for UDP streams, and as such, STUN keepalives **MUST** be used when an agent is a full ICE implementation and is communicating with a peer that supports ICE (lite or full). If the peer does not support ICE, the choice of a packet format for keepalives is a matter of local implementation. A format that allows packets to easily be sent in the absence of actual media content is

RECOMMENDED. Examples of formats that readily meet this goal are RTP No-Op [I-D.ietf-avt-rtp-no-op], and in cases where both sides support it, RTP comfort noise [RFC3389]. If the peer doesn't support any formats that are particularly well suited for keepalives, an agent SHOULD send RTP packets with an incorrect version number, or some other form of error that would cause them to be discarded by the peer.

If there has been no packet sent on the candidate pair ICE is using for a media component for Tr seconds (where packets include those defined for the component (RTP or RTCP) and previous keepalives), an agent MUST generate a keepalive on that pair. Tr SHOULD be configurable and SHOULD have a default of 15 seconds. Tr MUST NOT be configured to less than 15 seconds. Alternatively, if an agent has a dynamic way to discover the binding lifetimes of the intervening NATs, it can use that value to determine Tr. Administrators deploying ICE in more controlled networking environments SHOULD set Tr to the longest duration possible in their environment.

If STUN is being used for keepalives, a STUN Binding Indication is used [RFC5389]. The Indication MUST NOT utilize any authentication mechanism. It SHOULD contain the FINGERPRINT attribute to aid in demultiplexing, but SHOULD NOT contain any other attributes. It is used solely to keep the NAT bindings alive. The Binding Indication is sent using the same local and remote candidates that are being used for media. Though Binding Indications are used for keepalives, an agent MUST be prepared to receive a connectivity check as well. If a connectivity check is received, a response is generated as discussed in [RFC5389], but there is no impact on ICE processing otherwise.

An agent MUST begin the keepalive processing once ICE has selected candidates for usage with media, or media begins to flow, whichever happens first. Keepalives end once the session terminates or the media stream is removed.

10. Media Handling

10.1. Sending Media

Procedures for sending media differ for full and lite implementations.

10.1.1. Procedures for Full Implementations

Agents always send media using a candidate pair, called the selected candidate pair. An agent will send media to the remote candidate in

the selected pair (setting the destination address and port of the packet equal to that remote candidate), and will send it from the local candidate of the selected pair. When the local candidate is server or peer reflexive, media is originated from the base. Media sent from a relayed candidate is sent from the base through that TURN server, using procedures defined in [RFC5766].

If the local candidate is a relayed candidate, it is RECOMMENDED that an agent create a channel on the TURN server towards the remote candidate. This is done using the procedures for channel creation as defined in Section 11 of [RFC5766].

The selected pair for a component of a media stream is:

- o empty if the state of the check list for that media stream is Running, and there is no previous selected pair for that component due to an ICE restart
- o equal to the previous selected pair for a component of a media stream if the state of the check list for that media stream is Running, and there was a previous selected pair for that component due to an ICE restart
- o equal to the highest-priority nominated pair for that component in the valid list if the state of the check list is Completed

If the selected pair for at least one component of a media stream is empty, an agent MUST NOT send media for any component of that media stream. If the selected pair for each component of a media stream has a value, an agent MAY send media for all components of that media stream.

10.1.2. Procedures for Lite Implementations

A lite implementation MUST NOT send media until it has a Valid list that contains a candidate pair for each component of that media stream. Once that happens, the agent MAY begin sending media packets. To do that, it sends media to the remote candidate in the pair (setting the destination address and port of the packet equal to that remote candidate), and will send it from the local candidate.

10.1.3. Procedures for All Implementations

ICE has interactions with jitter buffer adaptation mechanisms. An RTP stream can begin using one candidate, and switch to another one, though this happens rarely with ICE. The newer candidate may result in RTP packets taking a different path through the network -- one with different delay characteristics. As discussed below, agents are

encouraged to re-adjust jitter buffers when there are changes in source or destination address of media packets. Furthermore, many audio codecs use the marker bit to signal the beginning of a talkspurt, for the purposes of jitter buffer adaptation. For such codecs, it is RECOMMENDED that the sender set the marker bit [RFC3550] when an agent switches transmission of media from one candidate pair to another.

10.2. Receiving Media

ICE implementations MUST be prepared to receive media on each component on any candidates provided for that component in the most recent offer/answer exchange (in the case of RTP, this would include both RTP and RTCP if candidates were provided for both).

It is RECOMMENDED that, when an agent receives an RTP packet with a new source or destination IP address for a particular media stream, that the agent re-adjust its jitter buffers.

RFC 3550 [RFC3550] describes an algorithm in Section 8.2 for detecting synchronization source (SSRC) collisions and loops. These algorithms are based, in part, on seeing different source transport addresses with the same SSRC. However, when ICE is used, such changes will sometimes occur as the media streams switch between candidates. An agent will be able to determine that a media stream is from the same peer as a consequence of the STUN exchange that proceeds media transmission. Thus, if there is a change in source transport address, but the media packets come from the same peer agent, this SHOULD NOT be treated as an SSRC collision.

11. Extensibility Considerations

This specification makes very specific choices about how both agents in a session coordinate to arrive at the set of candidate pairs that are selected for media. It is anticipated that future specifications will want to alter these algorithms, whether they are simple changes like timer tweaks or larger changes like a revamp of the priority algorithm. When such a change is made, providing interoperability between the two agents in a session is critical.

First, ICE provides the ice-options attribute. Each extension or change to ICE is associated with a token. When an agent supporting such an extension or change generates an offer or an answer, it MUST include the token for that extension in this attribute. This allows each side to know what the other side is doing. This attribute MUST NOT be present if the agent doesn't support any ICE extensions or changes.

One of the complications in achieving interoperability is that ICE relies on a distributed algorithm running on both agents to converge on an agreed set of candidate pairs. If the two agents run different algorithms, it can be difficult to guarantee convergence on the same candidate pairs. The regular nomination procedure described in Section 8 eliminates some of the tight coordination by delegating the selection algorithm completely to the controlling agent. Consequently, when a controlling agent is communicating with a peer that supports options it doesn't know about, the agent **MUST** run a regular nomination algorithm. When regular nomination is used, ICE will converge perfectly even when both agents use different pair prioritization algorithms. One of the keys to such convergence is triggered checks, which ensure that the nominated pair is validated by both agents. Consequently, any future ICE enhancements **MUST** preserve triggered checks.

ICE is also extensible to other media streams beyond RTP, and for transport protocols beyond UDP. Extensions to ICE for non-RTP media streams need to specify how many components they utilize, and assign component IDs to them, starting at 1 for the most important component ID. Specifications for new transport protocols must define how, if at all, various steps in the ICE processing differ from UDP.

12. Setting Ta and RTO

During the gathering phase of ICE (Section 4.1.1) and while ICE is performing connectivity checks (Section 7), an agent sends STUN and TURN transactions. These transactions are paced at a rate of one every Ta milliseconds, and utilize a specific RTO. This section describes how the values of Ta and RTO are computed. This computation depends on whether ICE is being used with a real-time media stream (such as RTP) or something else. When ICE is used for a stream with a known maximum bandwidth, the computation in Section 12.1 **MAY** be followed to rate-control the ICE exchanges. For all other streams, the computation in Section 12.2 **MUST** be followed.

12.1. RTP Media Streams

The values of RTO and Ta change during the lifetime of ICE processing. One set of values applies during the gathering phase, and the other, for connectivity checks.

The value of Ta **SHOULD** be configurable, and **SHOULD** have a default of:

For each media stream i:

$$Ta_i = (stun_packet_size / rtp_packet_size) * rtp_ptime$$

$$Ta = \text{MAX} (20\text{ms}, \frac{1}{k} \sum_{i=1}^k \frac{1}{Ta_i})$$

where k is the number of media streams. During the gathering phase, Ta is computed based on the number of media streams the agent has indicated in its offer or answer, and the RTP packet size and RTP ptime are those of the most preferred codec for each media stream. Once an offer and answer have been exchanged, the agent recomputes Ta to pace the connectivity checks. In that case, the value of Ta is based on the number of media streams that will actually be used in the session, and the RTP packet size and RTP ptime are those of the most preferred codec with which the agent will send.

In addition, the retransmission timer for the STUN transactions, RTO, defined in [RFC5389], SHOULD be configurable and during the gathering phase, SHOULD have a default of:

$$RTO = \text{MAX} (100\text{ms}, Ta * (\text{number of pairs}))$$

where the number of pairs refers to the number of pairs of candidates with STUN or TURN servers.

For connectivity checks, RTO SHOULD be configurable and SHOULD have a default of:

$$RTO = \text{MAX} (100\text{ms}, Ta * N * (\text{Num-Waiting} + \text{Num-In-Progress}))$$

where Num-Waiting is the number of checks in the check list in the Waiting state, and Num-In-Progress is the number of checks in the In-Progress state. Note that the RTO will be different for each transaction as the number of checks in the Waiting and In-Progress states change.

These formulas are aimed at causing STUN transactions to be paced at the same rate as media. This ensures that ICE will work properly

under the same network conditions needed to support the media as well. See Appendix B.1 for additional discussion and motivations. Because of this pacing, it will take a certain amount of time to obtain all of the server reflexive and relayed candidates. Implementations should be aware of the time required to do this, and if the application requires a time budget, limit the number of candidates that are gathered.

The formulas result in a behavior whereby an agent will send its first packet for every single connectivity check before performing a retransmit. This can be seen in the formulas for the RTO (which represents the retransmit interval). Those formulas scale with N, the number of checks to be performed. As a result of this, ICE maintains a nicely constant rate, but becomes more sensitive to packet loss. The loss of the first single packet for any connectivity check is likely to cause that pair to take a long time to be validated, and instead, a lower-priority check (but one for which there was no packet loss) is much more likely to complete first. This results in ICE performing sub-optimally, choosing lower-priority pairs over higher-priority pairs. Implementors should be aware of this consequence, but still should utilize the timer values described here.

12.2. Non-RTP Sessions

In cases where ICE is used to establish some kind of session that is not real time, and has no fixed rate associated with it that is known to work on the network in which ICE is deployed, Ta and RTO revert to more conservative values. Ta SHOULD be configurable, SHOULD have a default of 500 ms, and MUST NOT be configurable to be less than 500 ms.

In addition, the retransmission timer for the STUN transactions, RTO, SHOULD be configurable and during the gathering phase, SHOULD have a default of:

$$RTO = \text{MAX} (500\text{ms}, Ta * (\text{number of pairs}))$$

where the number of pairs refers to the number of pairs of candidates with STUN or TURN servers.

For connectivity checks, RTO SHOULD be configurable and SHOULD have a default of:

$$RTO = \text{MAX} (500\text{ms}, Ta * N * (\text{Num-Waiting} + \text{Num-In-Progress}))$$

13. Example

The example is based on the simplified topology of Figure 8.

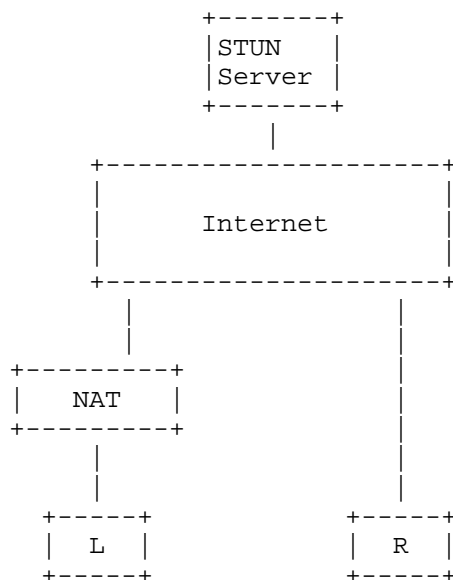


Figure 8: Example Topology

Two agents, L and R, are using ICE. Both are full-mode ICE implementations and use aggressive nomination when they are controlling. Both agents have a single IPv4 address. For agent L, it is 10.0.1.1 in private address space [RFC1918], and for agent R, 192.0.2.1 on the public Internet. Both are configured with the same STUN server (shown in this example for simplicity, although in practice the agents do not need to use the same STUN server), which is listening for STUN Binding requests at an IP address of 192.0.2.2 and port 3478. TURN servers are not used in this example. Agent L is behind a NAT, and agent R is on the public Internet. The NAT has an endpoint independent mapping property and an address dependent filtering property. The public side of the NAT has an IP address of 192.0.2.3.

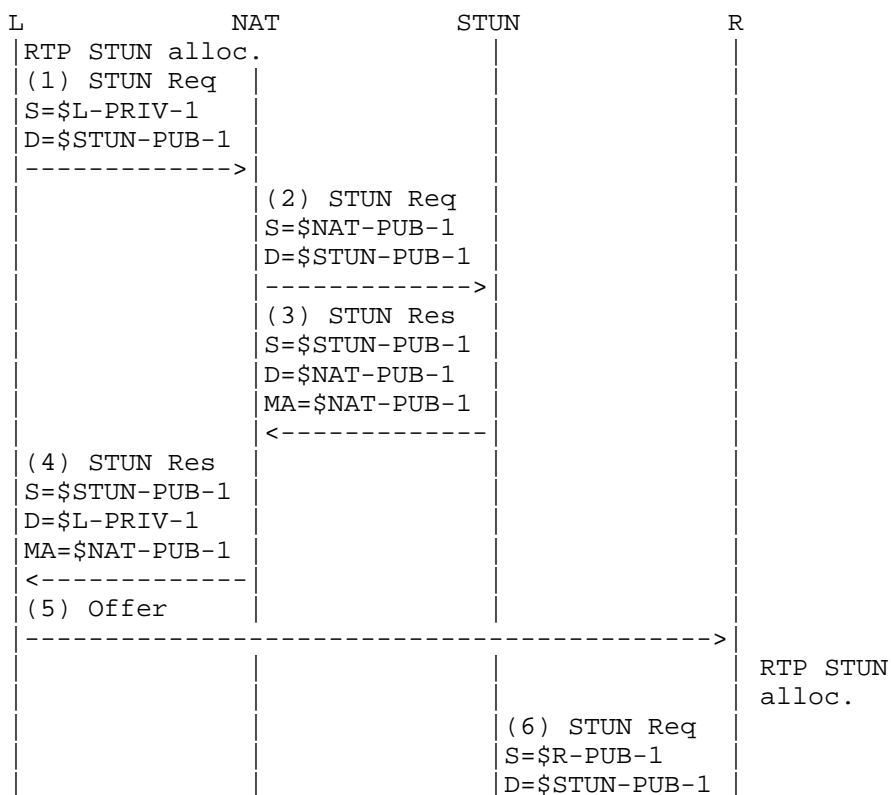
To facilitate understanding, transport addresses are listed using variables that have mnemonic names. The format of the name is entity-type-seqno, where entity refers to the entity whose IP address the transport address is on, and is one of "L", "R", "STUN", or "NAT". The type is either "PUB" for transport addresses that are public, and "PRIV" for transport addresses that are private.

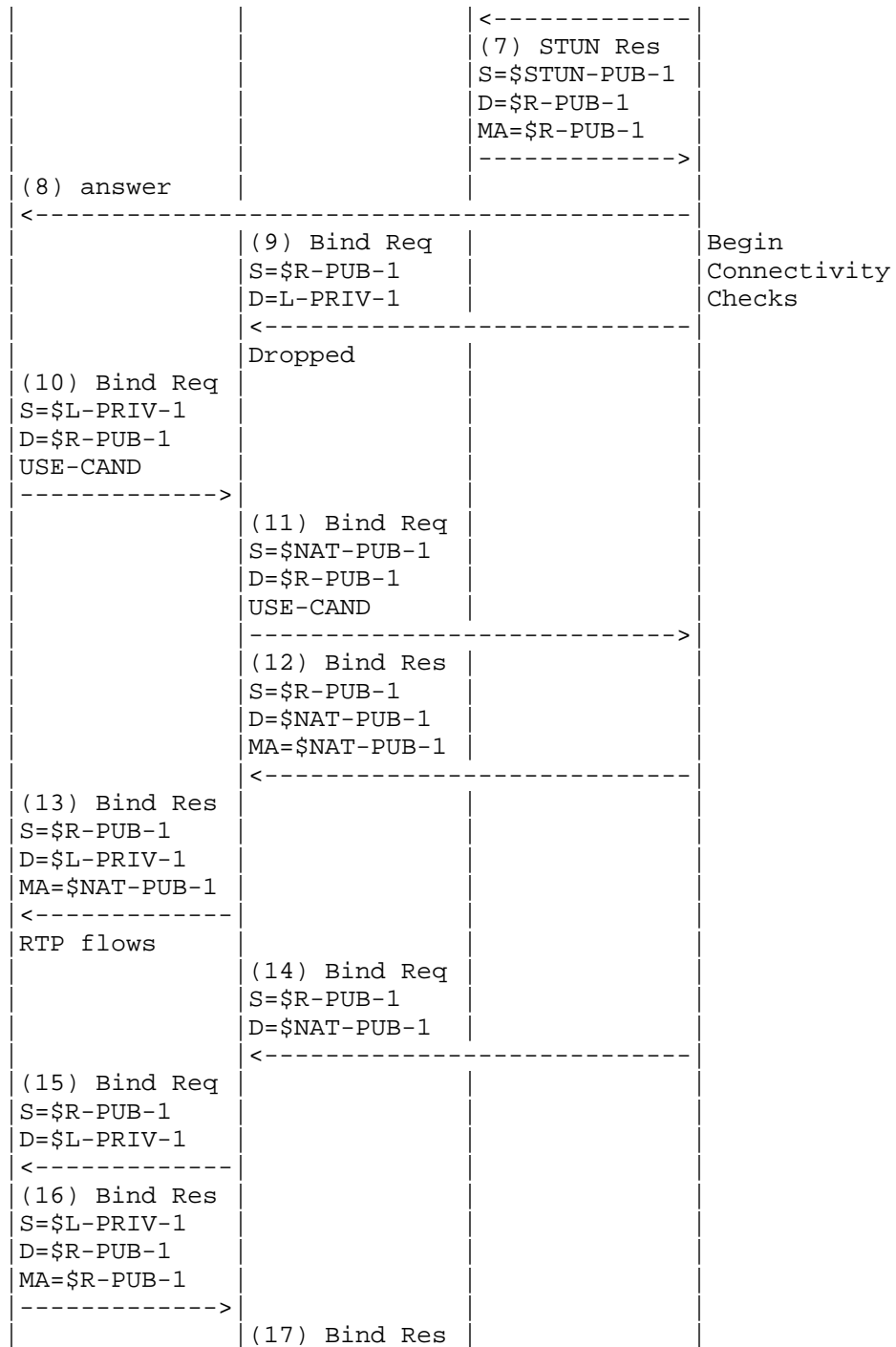
Finally, seq-no is a sequence number that is different for each transport address of the same type on a particular entity. Each variable has an IP address and port, denoted by varname.IP and varname.PORT, respectively, where varname is the name of the variable.

The STUN server has advertised transport address STUN-PUB-1 (which is 192.0.2.2:3478).

In the call flow itself, STUN messages are annotated with several attributes. The "S=" attribute indicates the source transport address of the message. The "D=" attribute indicates the destination transport address of the message. The "MA=" attribute is used in STUN Binding response messages and refers to the mapped address. "USE-CAND" implies the presence of the USE-CANDIDATE attribute.

The call flow examples omit STUN authentication operations and RTCP, and focus on RTP for a single media stream between two full implementations.





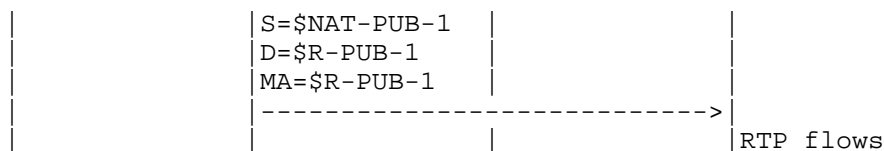


Figure 9: Example Flow

First, agent L obtains a host candidate from its local IP address (not shown), and from that, sends a STUN Binding request to the STUN server to get a server reflexive candidate (messages 1-4). Recall that the NAT has the address and port independent mapping property. Here, it creates a binding of NAT-PUB-1 for this UDP request, and this becomes the server reflexive candidate for RTP.

Agent L sets a type preference of 126 for the host candidate and 100 for the server reflexive. The local preference is 65535. Based on this, the priority of the host candidate is 2130706431 and for the server reflexive candidate is 1694498815. The host candidate is assigned a foundation of 1, and the server reflexive, a foundation of 2. These are sent to the peer in an offer.

This offer is received at agent R. Agent R will obtain a host candidate, and from it, obtain a server reflexive candidate (messages 6-7). Since R is not behind a NAT, this candidate is identical to its host candidate, and they share the same base. It therefore discards this redundant candidate and ends up with a single host candidate. With identical type and local preferences as L, the priority for this candidate is 2130706431. It chooses a foundation of 1 for its single candidate. The answerer's candidates are then sent to the offerer.

Since neither side indicated that it is lite, the agent that sent the offer that began ICE processing (agent L) becomes the controlling agent.

Agents L and R both pair up the candidates. They both initially have two pairs. However, agent L will prune the pair containing its server reflexive candidate, resulting in just one. At agent L, this pair has a local candidate of $\$L_PRIV_1$ and remote candidate of $\$R_PUB_1$, and has a candidate pair priority of $4.57566E+18$ (note that an implementation would represent this as a 64-bit integer so as not to lose precision). At agent R, there are two pairs. The highest priority has a local candidate of $\$R_PUB_1$ and remote candidate of $\$L_PRIV_1$ and has a priority of $4.57566E+18$, and the second has a local candidate of $\$R_PUB_1$ and remote candidate of $\$NAT_PUB_1$ and priority $3.63891E+18$.

Agent R begins its connectivity check (message 9) for the first pair (between the two host candidates). Since R is the controlled agent for this session, the check omits the USE-CANDIDATE attribute. The host candidate from agent L is private and behind a NAT, and thus this check won't be successful, because the packet cannot be routed from R to L.

When agent L gets the answer, it performs its one and only connectivity check (messages 10-13). It implements the aggressive nomination algorithm, and thus includes a USE-CANDIDATE attribute in this check. Since the check succeeds, agent L creates a new pair, whose local candidate is from the mapped address in the Binding response (NAT-PUB-1 from message 13) and whose remote candidate is the destination of the request (R-PUB-1 from message 10). This is added to the valid list. In addition, it is marked as selected since the Binding request contained the USE-CANDIDATE attribute. Since there is a selected candidate in the Valid list for the one component of this media stream, ICE processing for this stream moves into the Completed state. Agent L can now send media if it so chooses.

Soon after receipt of the STUN Binding request from agent L (message 11), agent R will generate its triggered check. This check happens to match the next one on its check list -- from its host candidate to agent L's server reflexive candidate. This check (messages 14-17) will succeed. Consequently, agent R constructs a new candidate pair using the mapped address from the response as the local candidate (R-PUB-1) and the destination of the request (NAT-PUB-1) as the remote candidate. This pair is added to the Valid list for that media stream. Since the check was generated in the reverse direction of a check that contained the USE-CANDIDATE attribute, the candidate pair is marked as selected. Consequently, processing for this stream moves into the Completed state, and agent R can also send media.

14. Security Considerations

There are several types of attacks possible in an ICE system. This section considers these attacks and their countermeasures. These countermeasures include:

- o Using ICE in conjunction with secure signaling techniques, such as SIPS.
- o Limiting the total number of connectivity checks to 100, and optionally limiting the number of candidates they'll accept in an offer or answer.

14.1. Attacks on Connectivity Checks

An attacker might attempt to disrupt the STUN connectivity checks. Ultimately, all of these attacks fool an agent into thinking something incorrect about the results of the connectivity checks. The possible false conclusions an attacker can try and cause are:

False Invalid: An attacker can fool a pair of agents into thinking a candidate pair is invalid, when it isn't. This can be used to cause an agent to prefer a different candidate (such as one injected by the attacker) or to disrupt a call by forcing all candidates to fail.

False Valid: An attacker can fool a pair of agents into thinking a candidate pair is valid, when it isn't. This can cause an agent to proceed with a session, but then not be able to receive any media.

False Peer Reflexive Candidate: An attacker can cause an agent to discover a new peer reflexive candidate, when it shouldn't have. This can be used to redirect media streams to a Denial-of-Service (DoS) target or to the attacker, for eavesdropping or other purposes.

False Valid on False Candidate: An attacker has already convinced an agent that there is a candidate with an address that doesn't actually route to that agent (for example, by injecting a false peer reflexive candidate or false server reflexive candidate). It must then launch an attack that forces the agents to believe that this candidate is valid.

If an attacker can cause a false peer reflexive candidate or false valid on a false candidate, it can launch any of the attacks described in [RFC5389].

To force the false invalid result, the attacker has to wait for the connectivity check from one of the agents to be sent. When it is, the attacker needs to inject a fake response with an unrecoverable error response, such as a 400. However, since the candidate is, in fact, valid, the original request may reach the peer agent, and result in a success response. The attacker needs to force this packet or its response to be dropped, through a DoS attack, layer 2 network disruption, or other technique. If it doesn't do this, the success response will also reach the originator, alerting it to a possible attack. Fortunately, this attack is mitigated completely through the STUN short-term credential mechanism. The attacker needs to inject a fake response, and in order for this response to be processed, the attacker needs the password. If the offer/answer

signaling is secured, the attacker will not have the password and its response will be discarded.

Forcing the fake valid result works in a similar way. The agent needs to wait for the Binding request from each agent, and inject a fake success response. The attacker won't need to worry about disrupting the actual response since, if the candidate is not valid, it presumably wouldn't be received anyway. However, like the fake invalid attack, this attack is mitigated by the STUN short-term credential mechanism in conjunction with a secure offer/answer exchange.

Forcing the false peer reflexive candidate result can be done either with fake requests or responses, or with replays. We consider the fake requests and responses case first. It requires the attacker to send a Binding request to one agent with a source IP address and port for the false candidate. In addition, the attacker must wait for a Binding request from the other agent, and generate a fake response with a XOR-MAPPED-ADDRESS attribute containing the false candidate. Like the other attacks described here, this attack is mitigated by the STUN message integrity mechanisms and secure offer/answer exchanges.

Forcing the false peer reflexive candidate result with packet replays is different. The attacker waits until one of the agents sends a check. It intercepts this request, and replays it towards the other agent with a faked source IP address. It must also prevent the original request from reaching the remote agent, either by launching a DoS attack to cause the packet to be dropped, or forcing it to be dropped using layer 2 mechanisms. The replayed packet is received at the other agent, and accepted, since the integrity check passes (the integrity check cannot and does not cover the source IP address and port). It is then responded to. This response will contain a XOR-MAPPED-ADDRESS with the false candidate, and will be sent to that false candidate. The attacker must then receive it and relay it towards the originator.

The other agent will then initiate a connectivity check towards that false candidate. This validation needs to succeed. This requires the attacker to force a false valid on a false candidate. Injecting of fake requests or responses to achieve this goal is prevented using the integrity mechanisms of STUN and the offer/answer exchange. Thus, this attack can only be launched through replays. To do that, the attacker must intercept the check towards this false candidate, and replay it towards the other agent. Then, it must intercept the response and replay that back as well.

This attack is very hard to launch unless the attacker is identified

by the fake candidate. This is because it requires the attacker to intercept and replay packets sent by two different hosts. If both agents are on different networks (for example, across the public Internet), this attack can be hard to coordinate, since it needs to occur against two different endpoints on different parts of the network at the same time.

If the attacker itself is identified by the fake candidate, the attack is easier to coordinate. However, if the media path is secured (e.g., using SRTP [RFC3711]), the attacker will not be able to play the media packets, but will only be able to discard them, effectively disabling the media stream for the call. However, this attack requires the agent to disrupt packets in order to block the connectivity check from reaching the target. In that case, if the goal is to disrupt the media stream, it's much easier to just disrupt it with the same mechanism, rather than attack ICE.

14.2. Attacks on Server Reflexive Address Gathering

ICE endpoints make use of STUN Binding requests for gathering server reflexive candidates from a STUN server. These requests are not authenticated in any way. As a consequence, there are numerous techniques an attacker can employ to provide the client with a false server reflexive candidate:

- o An attacker can compromise the DNS, causing DNS queries to return a rogue STUN server address. That server can provide the client with fake server reflexive candidates. This attack is mitigated by DNS security, though DNS-SEC is not required to address it.
- o An attacker that can observe STUN messages (such as an attacker on a shared network segment, like WiFi) can inject a fake response that is valid and will be accepted by the client.
- o An attacker can compromise a STUN server by means of a virus, and cause it to send responses with incorrect mapped addresses.

A false mapped address learned by these attacks will be used as a server reflexive candidate in the ICE exchange. For this candidate to actually be used for media, the attacker must also attack the connectivity checks, and in particular, force a false valid on a false candidate. This attack is very hard to launch if the false address identifies a fourth party (neither the offerer, answerer, nor attacker), since it requires attacking the checks generated by each agent in the session, and is prevented by SRTP if it identifies the attacker themselves.

If the attacker elects not to attack the connectivity checks, the

worst it can do is prevent the server reflexive candidate from being used. However, if the peer agent has at least one candidate that is reachable by the agent under attack, the STUN connectivity checks themselves will provide a peer reflexive candidate that can be used for the exchange of media. Peer reflexive candidates are generally preferred over server reflexive candidates. As such, an attack solely on the STUN address gathering will normally have no impact on a session at all.

14.3. Attacks on Relayed Candidate Gathering

An attacker might attempt to disrupt the gathering of relayed candidates, forcing the client to believe it has a false relayed candidate. Exchanges with the TURN server are authenticated using a long-term credential. Consequently, injection of fake responses or requests will not work. In addition, unlike Binding requests, Allocate requests are not susceptible to replay attacks with modified source IP addresses and ports, since the source IP address and port are not utilized to provide the client with its relayed candidate.

However, TURN servers are susceptible to DNS attacks, or to viruses aimed at the TURN server, for purposes of turning it into a zombie or rogue server. These attacks can be mitigated by DNS-SEC and through good box and software security on TURN servers.

Even if an attacker has caused the client to believe in a false relayed candidate, the connectivity checks cause such a candidate to be used only if they succeed. Thus, an attacker must launch a false valid on a false candidate, per above, which is a very difficult attack to coordinate.

14.4. Insider Attacks

In addition to attacks where the attacker is a third party trying to insert fake offers, answers, or stun messages, there are attacks possible with ICE when the attacker is an authenticated and valid participant in the ICE exchange.

14.4.1. STUN Amplification Attack

The STUN amplification attack is similar to the voice hammer. However, instead of voice packets being directed to the target, STUN connectivity checks are directed to the target. The attacker sends an offer with a large number of candidates, say, 50. The answerer receives the offer, and starts its checks, which are directed at the target, and consequently, never generate a response. The answerer will start a new connectivity check every T_a ms (say, $T_a=20$ ms). However, the retransmission timers are set to a large number due to

the large number of candidates. As a consequence, packets will be sent at an interval of one every T_a milliseconds, and then with increasing intervals after that. Thus, STUN will not send packets at a rate faster than media would be sent, and the STUN packets persist only briefly, until ICE fails for the session. Nonetheless, this is an amplification mechanism.

It is impossible to eliminate the amplification, but the volume can be reduced through a variety of heuristics. Agents SHOULD limit the total number of connectivity checks they perform to 100. Additionally, agents MAY limit the number of candidates they'll accept in an offer or answer.

Frequently, protocols that wish to avoid these kinds of attacks force the initiator to wait for a response prior to sending the next message. However, in the case of ICE, this is not possible. It is not possible to differentiate the following two cases:

- o There was no response because the initiator is being used to launch a DoS attack against an unsuspecting target that will not respond.
- o There was no response because the IP address and port are not reachable by the initiator.

In the second case, another check should be sent at the next opportunity, while in the former case, no further checks should be sent.

15. STUN Extensions

15.1. New Attributes

This specification defines four new attributes, PRIORITY, USE-CANDIDATE, ICE-CONTROLLED, and ICE-CONTROLLING.

The PRIORITY attribute indicates the priority that is to be associated with a peer reflexive candidate, should one be discovered by this check. It is a 32-bit unsigned integer, and has an attribute value of 0x0024.

The USE-CANDIDATE attribute indicates that the candidate pair resulting from this check should be used for transmission of media. The attribute has no content (the Length field of the attribute is zero); it serves as a flag. It has an attribute value of 0x0025.

The ICE-CONTROLLED attribute is present in a Binding request and

indicates that the client believes it is currently in the controlled role. The content of the attribute is a 64-bit unsigned integer in network byte order, which contains a random number used for tie-breaking of role conflicts.

The ICE-CONTROLLING attribute is present in a Binding request and indicates that the client believes it is currently in the controlling role. The content of the attribute is a 64-bit unsigned integer in network byte order, which contains a random number used for tie-breaking of role conflicts.

15.2. New Error Response Codes

This specification defines a single error response code:

487 (Role Conflict): The Binding request contained either the ICE-CONTROLLING or ICE-CONTROLLED attribute, indicating a role that conflicted with the server. The server ran a tie-breaker based on the tie-breaker value in the request and determined that the client needs to switch roles.

16. Operational Considerations

This section discusses issues relevant to network operators looking to deploy ICE.

16.1. NAT and Firewall Types

ICE was designed to work with existing NAT and firewall equipment. Consequently, it is not necessary to replace or reconfigure existing firewall and NAT equipment in order to facilitate deployment of ICE. Indeed, ICE was developed to be deployed in environments where the Voice over IP (VoIP) operator has no control over the IP network infrastructure, including firewalls and NAT.

That said, ICE works best in environments where the NAT devices are "behave" compliant, meeting the recommendations defined in [RFC4787] and [RFC5382]. In networks with behave-compliant NAT, ICE will work without the need for a TURN server, thus improving voice quality, decreasing call setup times, and reducing the bandwidth demands on the network operator.

16.2. Bandwidth Requirements

Deployment of ICE can have several interactions with available network capacity that operators should take into consideration.

16.2.1. STUN and TURN Server Capacity Planning

First and foremost, ICE makes use of TURN and STUN servers, which would typically be located in the network operator's data centers. The STUN servers require relatively little bandwidth. For each component of each media stream, there will be one or more STUN transactions from each client to the STUN server. In a basic voice-only IPv4 VoIP deployment, there will be four transactions per call (one for RTP and one for RTCP, for both caller and callee). Each transaction is a single request and a single response, the former being 20 bytes long, and the latter, 28. Consequently, if a system has N users, and each makes four calls in a busy hour, this would require $N \times 1.7\text{bps}$. For one million users, this is 1.7 Mbps, a very small number (relatively speaking).

TURN traffic is more substantial. The TURN server will see traffic volume equal to the STUN volume (indeed, if TURN servers are deployed, there is no need for a separate STUN server), in addition to the traffic for the actual media traffic. The amount of calls requiring TURN for media relay is highly dependent on network topologies, and can and will vary over time. In a network with 100% behave-compliant NAT, it is exactly zero. At time of writing, large-scale consumer deployments were seeing between 5 and 10 percent of calls requiring TURN servers. Considering a voice-only deployment using G.711 (so 80 kbps in each direction), with .2 erlangs during the busy hour, this is $N \times 3.2\text{ kbps}$. For a population of one million users, this is 3.2 Gbps, assuming a 10% usage of TURN servers.

16.2.2. Gathering and Connectivity Checks

The process of gathering of candidates and performing of connectivity checks can be bandwidth intensive. ICE has been designed to pace both of these processes. The gathering phase and the connectivity check phase are meant to generate traffic at roughly the same bandwidth as the media traffic itself. This was done to ensure that, if a network is designed to support multimedia traffic of a certain type (voice, video, or just text), it will have sufficient capacity to support the ICE checks for that media. Of course, the ICE checks will cause a marginal increase in the total utilization; however, this will typically be an extremely small increase.

Congestion due to the gathering and check phases has proven to be a problem in deployments that did not utilize pacing. Typically, access links became congested as the endpoints flooded the network with checks as fast as they can send them. Consequently, network operators should make sure that their ICE implementations support the pacing feature. Though this pacing does increase call setup times, it makes ICE network friendly and easier to deploy.

16.2.3. Keepalives

STUN keepalives (in the form of STUN Binding Indications) are sent in the middle of a media session. However, they are sent only in the absence of actual media traffic. In deployments that are not utilizing Voice Activity Detection (VAD), the keepalives are never used and there is no increase in bandwidth usage. When VAD is being used, keepalives will be sent during silence periods. This involves a single packet every 15-20 seconds, far less than the packet every 20-30 ms that is sent when there is voice. Therefore, keepalives don't have any real impact on capacity planning.

16.3. ICE and ICE-lite

Deployments utilizing a mix of ICE and ICE-lite interoperate perfectly. They have been explicitly designed to do so, without loss of function.

However, ICE-lite can only be deployed in limited use cases. Those cases, and the caveats involved in doing so, are documented in Appendix A.

16.4. Troubleshooting and Performance Management

ICE utilizes end-to-end connectivity checks, and places much of the processing in the endpoints. This introduces a challenge to the network operator -- how can they troubleshoot ICE deployments? How can they know how ICE is performing?

ICE has built-in features to help deal with these problems. SIP servers on the signaling path, typically deployed in the data centers of the network operator, will see the contents of the offer/answer exchanges that convey the ICE parameters. These parameters include the type of each candidate (host, server reflexive, or relayed), along with their related addresses. Once ICE processing has completed, an updated offer/answer exchange takes place, signaling the selected address (and its type). This updated re-INVITE is performed exactly for the purposes of educating network equipment (such as a diagnostic tool attached to a SIP server) about the results of ICE processing.

As a consequence, through the logs generated by the SIP server, a network operator can observe what types of candidates are being used for each call, and what address was selected by ICE. This is the primary information that helps evaluate how ICE is performing.

16.5. Endpoint Configuration

ICE relies on several pieces of data being configured into the endpoints. This configuration data includes timers, credentials for TURN servers, and hostnames for STUN and TURN servers. ICE itself does not provide a mechanism for this configuration. Instead, it is assumed that this information is attached to whatever mechanism is used to configure all of the other parameters in the endpoint. For SIP phones, standard solutions such as the configuration framework [RFC6080] have been defined.

17. IANA Considerations

The original ICE specification registered four new STUN attributes, and one new STUN error response. The STUN attributes and error response are reproduced here.

17.1. STUN Attributes

IANA has registered four STUN attributes:

```
0x0024 PRIORITY
0x0025 USE-CANDIDATE
0x8029 ICE-CONTROLLED
0x802A ICE-CONTROLLING
```

17.2. STUN Error Responses

IANA has registered following STUN error response code:

```
487    Role Conflict: The client asserted an ICE role (controlling or
        controlled) that is in conflict with the role of the server.
```

18. IAB Considerations

The IAB has studied the problem of "Unilateral Self-Address Fixing", which is the general process by which a agent attempts to determine its address in another realm on the other side of a NAT through a collaborative protocol reflection mechanism [RFC3424]. ICE is an example of a protocol that performs this type of function. Interestingly, the process for ICE is not unilateral, but bilateral, and the difference has a significant impact on the issues raised by IAB. Indeed, ICE can be considered a B-SAF (Bilateral Self-Address Fixing) protocol, rather than an UNSAF protocol. Regardless, the IAB

has mandated that any protocols developed for this purpose document a specific set of considerations. This section meets those requirements.

18.1. Problem Definition

>From RFC 3424, any UNSAF proposal must provide:

Precise definition of a specific, limited-scope problem that is to be solved with the UNSAF proposal. A short-term fix should not be generalized to solve other problems; this is why "short-term fixes usually aren't".

The specific problems being solved by ICE are:

Provide a means for two peers to determine the set of transport addresses that can be used for communication.

Provide a means for a agent to determine an address that is reachable by another peer with which it wishes to communicate.

18.2. Exit Strategy

>From RFC 3424, any UNSAF proposal must provide:

Description of an exit strategy/transition plan. The better short-term fixes are the ones that will naturally see less and less use as the appropriate technology is deployed.

ICE itself doesn't easily get phased out. However, it is useful even in a globally connected Internet, to serve as a means for detecting whether a router failure has temporarily disrupted connectivity, for example. ICE also helps prevent certain security attacks that have nothing to do with NAT. However, what ICE does is help phase out other UNSAF mechanisms. ICE effectively selects amongst those mechanisms, prioritizing ones that are better, and deprioritizing ones that are worse. Local IPv6 addresses can be preferred. As NATs begin to dissipate as IPv6 is introduced, server reflexive and relayed candidates (both forms of UNSAF addresses) simply never get used, because higher-priority connectivity exists to the native host candidates. Therefore, the servers get used less and less, and can eventually be remove when their usage goes to zero.

Indeed, ICE can assist in the transition from IPv4 to IPv6. It can be used to determine whether to use IPv6 or IPv4 when two dual-stack hosts communicate with SIP (IPv6 gets used). It can also allow a network with both 6to4 and native v6 connectivity to determine which address to use when communicating with a peer.

18.3. Brittleness Introduced by ICE

>From RFC 3424, any UNSAF proposal must provide:

Discussion of specific issues that may render systems more "brittle". For example, approaches that involve using data at multiple network layers create more dependencies, increase debugging challenges, and make it harder to transition.

ICE actually removes brittleness from existing UNSAF mechanisms. In particular, classic STUN (as described in RFC 3489 [RFC3489]) has several points of brittleness. One of them is the discovery process that requires an agent to try to classify the type of NAT it is behind. This process is error-prone. With ICE, that discovery process is simply not used. Rather than unilaterally assessing the validity of the address, its validity is dynamically determined by measuring connectivity to a peer. The process of determining connectivity is very robust.

Another point of brittleness in classic STUN and any other unilateral mechanism is its absolute reliance on an additional server. ICE makes use of a server for allocating unilateral addresses, but allows agents to directly connect if possible. Therefore, in some cases, the failure of a STUN server would still allow for a call to progress when ICE is used.

Another point of brittleness in classic STUN is that it assumes that the STUN server is on the public Internet. Interestingly, with ICE, that is not necessary. There can be a multitude of STUN servers in a variety of address realms. ICE will discover the one that has provided a usable address.

The most troubling point of brittleness in classic STUN is that it doesn't work in all network topologies. In cases where there is a shared NAT between each agent and the STUN server, traditional STUN may not work. With ICE, that restriction is removed.

Classic STUN also introduces some security considerations. Fortunately, those security considerations are also mitigated by ICE.

Consequently, ICE serves to repair the brittleness introduced in classic STUN, and does not introduce any additional brittleness into the system.

The penalty of these improvements is that ICE increases session establishment times.

18.4. Requirements for a Long-Term Solution

From RFC 3424, any UNSAF proposal must provide:

... requirements for longer term, sound technical solutions -- contribute to the process of finding the right longer term solution.

Our conclusions from RFC 3489 remain unchanged. However, we feel ICE actually helps because we believe it can be part of the long-term solution.

18.5. Issues with Existing NAPT Boxes

From RFC 3424, any UNSAF proposal must provide:

Discussion of the impact of the noted practical issues with existing, deployed NA[P]Ts and experience reports.

A number of NAT boxes are now being deployed into the market that try to provide "generic" ALG functionality. These generic ALGs hunt for IP addresses, either in text or binary form within a packet, and rewrite them if they match a binding. This interferes with classic STUN. However, the update to STUN [RFC5389] uses an encoding that hides these binary addresses from generic ALGs.

Existing NAPT boxes have non-deterministic and typically short expiration times for UDP-based bindings. This requires implementations to send periodic keepalives to maintain those bindings. ICE uses a default of 15 s, which is a very conservative estimate. Eventually, over time, as NAT boxes become compliant to behave [RFC4787], this minimum keepalive will become deterministic and well-known, and the ICE timers can be adjusted. Having a way to discover and control the minimum keepalive interval would be far better still.

19. Changes from RFC 5245

Following is the list of changes from RFC 5245

- o The specification was generalized to be more usable with any protocol and the parts that are specific to SIP and SDP were moved to a SIP/SDP usage document [I-D.petithuguenin-mmusic-ice-sip-sdp].
- o Default candidates, multiple components, ICE mismatch detection, subsequent offer/answer, and role conflict resolution were made

optional since they are not needed with every protocol using ICE.

- o With IPv6, the precedence rules of RFC 6724 are used instead of the obsoleted RFC 3483 and using address preferences provided by the host operating system is recommended.
- o Candidate gathering rules regarding loopback addresses and IPv6 addresses were clarified.

20. Acknowledgements

Most of the text in this document comes from the original ICE specification, RFC 5245. The authors would like to thank everyone who has contributed to that document.

21. References

21.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

21.2. Informative References

- [RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.
- [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.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model

with Session Description Protocol (SDP)", RFC 3264, June 2002.

- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- [RFC3235] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002.
- [RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.
- [RFC3102] Borella, M., Lo, J., Grabelsky, D., and G. Montenegro, "Realm Specific IP: Framework", RFC 3102, October 2001.
- [RFC3103] Borella, M., Grabelsky, D., Lo, J., and K. Taniguchi, "Realm Specific IP: Protocol Specification", RFC 3103, October 2001.
- [RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.
- [RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.
- [RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, September 2004.
- [RFC4038] Shin, M-K., Hong, Y-G., Hagino, J., Savola, P., and E. Castro, "Application Aspects of IPv6 Transition", RFC 4038, March 2005.
- [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.

- [RFC4092] Camarillo, G. and J. Rosenberg, "Usage of the Session Description Protocol (SDP) Alternative Network Address Types (ANAT) Semantics in the Session Initiation Protocol (SIP)", RFC 4092, June 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [I-D.ietf-avt-rtp-no-op]
Andreasen, F., "A No-Op Payload Format for RTP",
draft-ietf-avt-rtp-no-op-04 (work in progress), May 2007.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.
- [RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.
- [RFC6080] Petrie, D. and S. Channabasappa, "A Framework for Session Initiation Protocol User Agent Profile Delivery", RFC 6080, March 2011.
- [RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

[I-D.petithuguenin-mmusic-ice-sip-sdp]

Petit-Huguenin, M. and A. Keraenen, "Using Interactive Connectivity Establishment (ICE) with Session Description Protocol (SDP) offer/answer and Session Initiation Protocol (SIP)", draft-petithuguenin-mmusic-ice-sip-sdp-00 (work in progress), February 2013.

Appendix A. Lite and Full Implementations

ICE allows for two types of implementations. A full implementation supports the controlling and controlled roles in a session, and can also perform address gathering. In contrast, a lite implementation is a minimalist implementation that does little but respond to STUN checks.

Because ICE requires both endpoints to support it in order to bring benefits to either endpoint, incremental deployment of ICE in a network is more complicated. Many sessions involve an endpoint that is, by itself, not behind a NAT and not one that would worry about NAT traversal. A very common case is to have one endpoint that requires NAT traversal (such as a VoIP hard phone or soft phone) make a call to one of these devices. Even if the phone supports a full ICE implementation, ICE won't be used at all if the other device doesn't support it. The lite implementation allows for a low-cost entry point for these devices. Once they support the lite implementation, full implementations can connect to them and get the full benefits of ICE.

Consequently, a lite implementation is only appropriate for devices that will **always** be connected to the public Internet and have a public IP address at which it can receive packets from any correspondent. ICE will not function when a lite implementation is placed behind a NAT.

ICE allows a lite implementation to have a single IPv4 host candidate and several IPv6 addresses. In that case, candidate pairs are selected by the controlling agent using a static algorithm, such as the one in RFC 6724, which is recommended by this specification. However, static mechanisms for address selection are always prone to error, since they cannot ever reflect the actual topology and can never provide actual guarantees on connectivity. They are always heuristics. Consequently, if an agent is implementing ICE just to select between its IPv4 and IPv6 addresses, and none of its IP addresses are behind NAT, usage of full ICE is still RECOMMENDED in order to provide the most robust form of address selection possible.

It is important to note that the lite implementation was added to

this specification to provide a stepping stone to full implementation. Even for devices that are always connected to the public Internet with just a single IPv4 address, a full implementation is preferable if achievable. A full implementation will reduce call setup times, since ICE's aggressive mode can be used. Full implementations also obtain the security benefits of ICE unrelated to NAT traversal; in particular, the voice hammer attack described in Section 14 is prevented only for full implementations, not lite. Finally, it is often the case that a device that finds itself with a public address today will be placed in a network tomorrow where it will be behind a NAT. It is difficult to definitively know, over the lifetime of a device or product, that it will always be used on the public Internet. Full implementation provides assurance that communications will always work.

Appendix B. Design Motivations

ICE contains a number of normative behaviors that may themselves be simple, but derive from complicated or non-obvious thinking or use cases that merit further discussion. Since these design motivations are not necessary to understand for purposes of implementation, they are discussed here in an appendix to the specification. This section is non-normative.

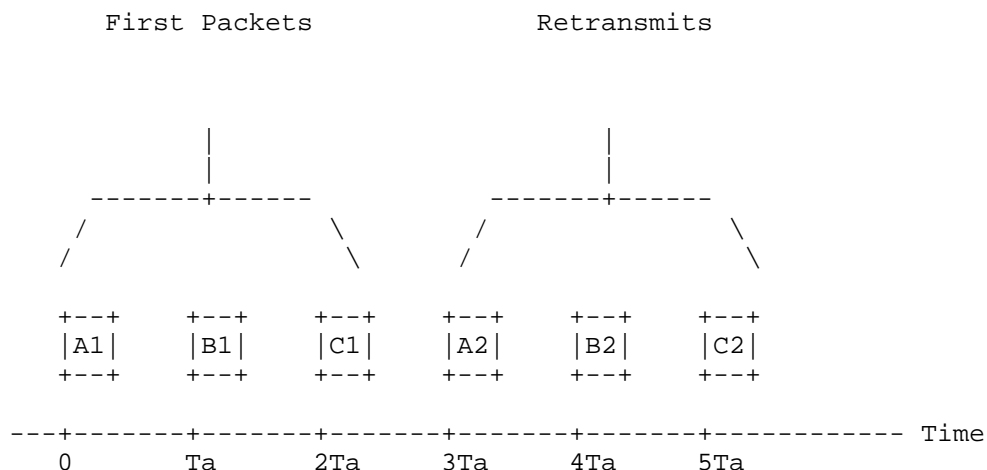
B.1. Pacing of STUN Transactions

STUN transactions used to gather candidates and to verify connectivity are paced out at an approximate rate of one new transaction every T_a milliseconds. Each transaction, in turn, has a retransmission timer RTO that is a function of T_a as well. Why are these transactions paced, and why are these formulas used?

Sending of these STUN requests will often have the effect of creating bindings on NAT devices between the client and the STUN servers. Experience has shown that many NAT devices have upper limits on the rate at which they will create new bindings. Experiments have shown that once every 20 ms is well supported, but not much lower than that. This is why T_a has a lower bound of 20 ms. Furthermore, transmission of these packets on the network makes use of bandwidth and needs to be rate limited by the agent. Deployments based on earlier draft versions of this document tended to overload rate-constrained access links and perform poorly overall, in addition to negatively impacting the network. As a consequence, the pacing ensures that the NAT device does not get overloaded and that traffic is kept at a reasonable rate.

The definition of a "reasonable" rate is that STUN should not use

more bandwidth than the RTP itself will use, once media starts flowing. The formula for T_a is designed so that, if a STUN packet were sent every T_a seconds, it would consume the same amount of bandwidth as RTP packets, summed across all media streams. Of course, STUN has retransmits, and the desire is to pace those as well. For this reason, RTO is set such that the first retransmit on the first transaction happens just as the first STUN request on the last transaction occurs. Pictorially:



In this picture, there are three transactions that will be sent (for example, in the case of candidate gathering, there are three host candidate/STUN server pairs). These are transactions A, B, and C. The retransmit timer is set so that the first retransmission on the first transaction (packet A2) is sent at time $3T_a$.

Subsequent retransmits after the first will occur even less frequently than T_a milliseconds apart, since STUN uses an exponential back-off on its retransmissions.

B.2. Candidates with Multiple Bases

Section 4.1.3 talks about eliminating candidates that have the same transport address and base. However, candidates with the same transport addresses but different bases are not redundant. When can an agent have two candidates that have the same IP address and port, but different bases? Consider the topology of Figure 10:

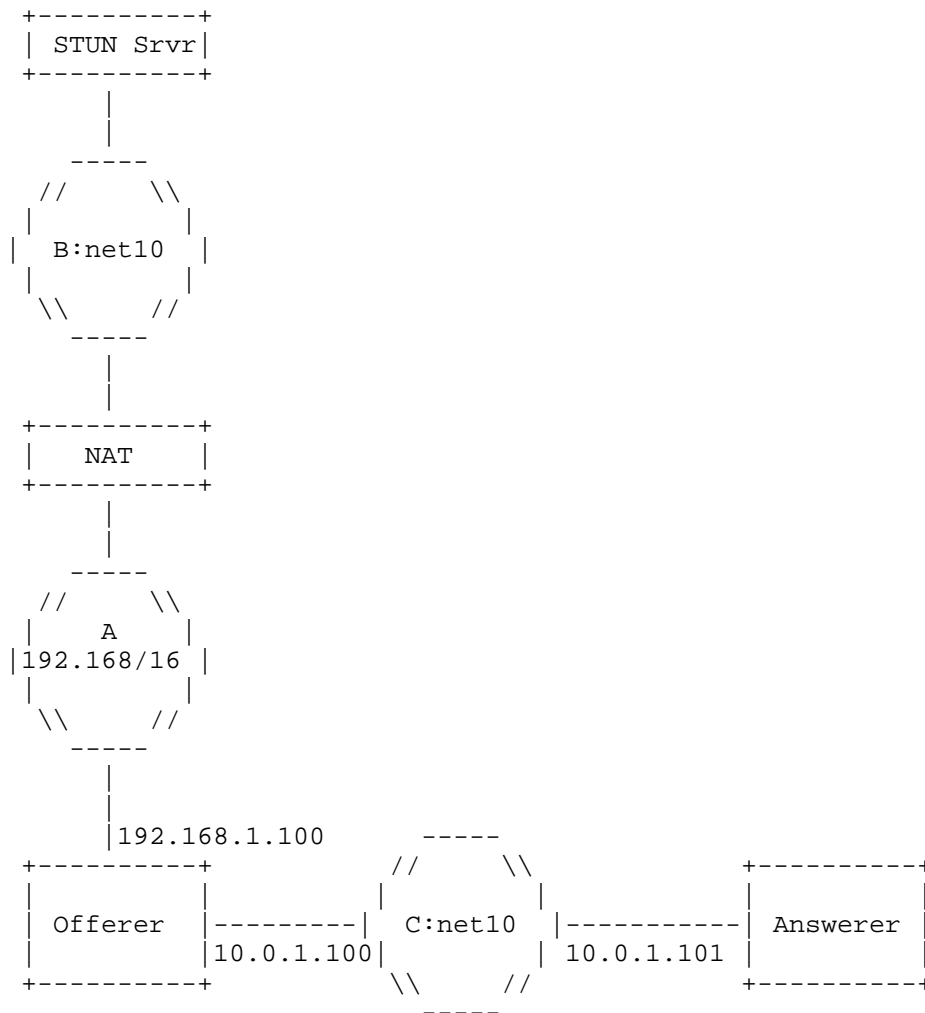


Figure 10: Identical Candidates with Different Bases

In this case, the offerer is multi-homed. It has one IP address, 10.0.1.100, on network C, which is a net 10 private network. The answerer is on this same network. The offerer is also connected to network A, which is 192.168/16. The offerer has an IP address of 192.168.1.100 on this network. There is a NAT on this network, natting into network B, which is another net 10 private network, but not connected to network C. There is a STUN server on network B.

The offerer obtains a host candidate on its IP address on network C (10.0.1.100:2498) and a host candidate on its IP address on network A (192.168.1.100:3344). It performs a STUN query to its configured STUN server from 192.168.1.100:3344. This query passes through the NAT, which happens to assign the binding 10.0.1.100:2498. The STUN server reflects this in the STUN Binding response. Now, the offerer has obtained a server reflexive candidate with a transport address that is identical to a host candidate (10.0.1.100:2498). However, the server reflexive candidate has a base of 192.168.1.100:3344, and the host candidate has a base of 10.0.1.100:2498.

B.3. Purpose of the Related Address and Related Port Attributes

The candidate attribute contains two values that are not used at all by ICE itself -- related address and related port. Why are they present?

There are two motivations for its inclusion. The first is diagnostic. It is very useful to know the relationship between the different types of candidates. By including it, an agent can know which relayed candidate is associated with which reflexive candidate, which in turn is associated with a specific host candidate. When checks for one candidate succeed and not for others, this provides useful diagnostics on what is going on in the network.

The second reason has to do with off-path Quality of Service (QoS) mechanisms. When ICE is used in environments such as PacketCable 2.0, proxies will, in addition to performing normal SIP operations, inspect the SDP in SIP messages, and extract the IP address and port for media traffic. They can then interact, through policy servers, with access routers in the network, to establish guaranteed QoS for the media flows. This QoS is provided by classifying the RTP traffic based on 5-tuple, and then providing it a guaranteed rate, or marking its Diffserv codepoints appropriately. When a residential NAT is present, and a relayed candidate gets selected for media, this relayed candidate will be a transport address on an actual TURN server. That address says nothing about the actual transport address in the access router that would be used to classify packets for QoS treatment. Rather, the server reflexive candidate towards the TURN server is needed. By carrying the translation in the SDP, the proxy can use that transport address to request QoS from the access router.

B.4. Importance of the STUN Username

ICE requires the usage of message integrity with STUN using its short-term credential functionality. The actual short-term credential is formed by exchanging username fragments in the offer/answer exchange. The need for this mechanism goes beyond just

security; it is actually required for correct operation of ICE in the first place.

Consider agents L, R, and Z. L and R are within private enterprise 1, which is using 10.0.0.0/8. Z is within private enterprise 2, which is also using 10.0.0.0/8. As it turns out, R and Z both have IP address 10.0.1.1. L sends an offer to Z. Z, in its answer, provides L with its host candidates. In this case, those candidates are 10.0.1.1:8866 and 10.0.1.1:8877. As it turns out, R is in a session at that same time, and is also using 10.0.1.1:8866 and 10.0.1.1:8877 as host candidates. This means that R is prepared to accept STUN messages on those ports, just as Z is. L will send a STUN request to 10.0.1.1:8866 and another to 10.0.1.1:8877. However, these do not go to Z as expected. Instead, they go to R! If R just replied to them, L would believe it has connectivity to Z, when in fact it has connectivity to a completely different user, R. To fix this, the STUN short-term credential mechanisms are used. The username fragments are sufficiently random that it is highly unlikely that R would be using the same values as Z. Consequently, R would reject the STUN request since the credentials were invalid. In essence, the STUN username fragments provide a form of transient host identifiers, bound to a particular offer/answer session.

An unfortunate consequence of the non-uniqueness of IP addresses is that, in the above example, R might not even be an ICE agent. It could be any host, and the port to which the STUN packet is directed could be any ephemeral port on that host. If there is an application listening on this socket for packets, and it is not prepared to handle malformed packets for whatever protocol is in use, the operation of that application could be affected. Fortunately, since the ports exchanged in offer/answer are ephemeral and usually drawn from the dynamic or registered range, the odds are good that the port is not used to run a server on host R, but rather is the agent side of some protocol. This decreases the probability of hitting an allocated port, due to the transient nature of port usage in this range. However, the possibility of a problem does exist, and network deployers should be prepared for it. Note that this is not a problem specific to ICE; stray packets can arrive at a port at any time for any type of protocol, especially ones on the public Internet. As such, this requirement is just restating a general design guideline for Internet applications -- be prepared for unknown packets on any port.

B.5. The Candidate Pair Priority Formula

The priority for a candidate pair has an odd form. It is:

$$\text{pair priority} = 2^{32} * \text{MIN}(G,D) + 2 * \text{MAX}(G,D) + (G > D ? 1 : 0)$$

Why is this? When the candidate pairs are sorted based on this value, the resulting sorting has the MAX/MIN property. This means that the pairs are first sorted based on decreasing value of the minimum of the two priorities. For pairs that have the same value of the minimum priority, the maximum priority is used to sort amongst them. If the max and the min priorities are the same, the controlling agent's priority is used as the tie-breaker in the last part of the expression. The factor of 2^{32} is used since the priority of a single candidate is always less than 2^{32} , resulting in the pair priority being a "concatenation" of the two component priorities. This creates the MAX/MIN sorting. MAX/MIN ensures that, for a particular agent, a lower-priority candidate is never used until all higher-priority candidates have been tried.

B.6. Why Are Keepalives Needed?

Once media begins flowing on a candidate pair, it is still necessary to keep the bindings alive at intermediate NATs for the duration of the session. Normally, the media stream packets themselves (e.g., RTP) meet this objective. However, several cases merit further discussion. Firstly, in some RTP usages, such as SIP, the media streams can be "put on hold". This is accomplished by using the SDP "sendonly" or "inactive" attributes, as defined in RFC 3264 [RFC3264]. RFC 3264 directs implementations to cease transmission of media in these cases. However, doing so may cause NAT bindings to timeout, and media won't be able to come off hold.

Secondly, some RTP payload formats, such as the payload format for text conversation [RFC4103], may send packets so infrequently that the interval exceeds the NAT binding timeouts.

Thirdly, if silence suppression is in use, long periods of silence may cause media transmission to cease sufficiently long for NAT bindings to time out.

For these reasons, the media packets themselves cannot be relied upon. ICE defines a simple periodic keepalive utilizing STUN Binding indications. This makes its bandwidth requirements highly predictable, and thus amenable to QoS reservations.

B.7. Why Prefer Peer Reflexive Candidates?

Section 4.1.2 describes procedures for computing the priority of candidate based on its type and local preferences. That section requires that the type preference for peer reflexive candidates always be higher than server reflexive. Why is that? The reason has

to do with the security considerations in Section 14. It is much easier for an attacker to cause an agent to use a false server reflexive candidate than it is for an attacker to cause an agent to use a false peer reflexive candidate. Consequently, attacks against address gathering with Binding requests are thwarted by ICE by preferring the peer reflexive candidates.

B.8. Why Are Binding Indications Used for Keepalives?

Media keepalives are described in Section 9. These keepalives make use of STUN when both endpoints are ICE capable. However, rather than using a Binding request transaction (which generates a response), the keepalives use an Indication. Why is that?

The primary reason has to do with network QoS mechanisms. Once media begins flowing, network elements will assume that the media stream has a fairly regular structure, making use of periodic packets at fixed intervals, with the possibility of jitter. If an agent is sending media packets, and then receives a Binding request, it would need to generate a response packet along with its media packets. This will increase the actual bandwidth requirements for the 5-tuple carrying the media packets, and introduce jitter in the delivery of those packets. Analysis has shown that this is a concern in certain layer 2 access networks that use fairly tight packet schedulers for media.

Additionally, using a Binding Indication allows integrity to be disabled, allowing for better performance. This is useful for large-scale endpoints, such as PSTN gateways and SBCs.

Authors' Addresses

Ari Keranen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: ari.keranen@ericsson.com

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

Email: jdrosen@jdrosen.net
URI: <http://www.jdrosen.net>

MMUSIC Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2014

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

Multiplexing Negotiation Using Session Description Protocol (SDP) Port
Numbers
draft-ietf-mmusic-sdp-bundle-negotiation-05.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 April 17, 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	4
3. Conventions	5
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	7
6.4.1. General	7
6.4.2. Request BUNDLE address selection	8
6.4.3. Bundle Address Synchronization (BAS)	8
6.4.4. Adding a media description to a BUNDLE group	8
6.4.5. Moving A Media Description Out Of A BUNDLE Group	9
6.4.6. Disabling A Media Description In A BUNDLE Group	9
6.5. SDP Answerer Procedures	9
6.5.1. Offerer Bundle Address Selection	10
6.5.2. Answerer Bundle Address Selection	10
6.5.3. Moving A Media Description Out Of A BUNDLE Group	10
6.5.4. Rejecting A Media Description In A BUNDLE Group	11
7. Single vs Multiple RTP Sessions	11
7.1. General	11
7.2. Single RTP Session	11
8. Usage With ICE	12
8.1. General	12
8.2. Candidates	12
8.3. Candidates	12
9. Security Considerations	12
10. Examples	13
10.1. Example: Bundle Address Selection	13
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	19
11. IANA Considerations	20
12. Acknowledgements	20
13. Change Log	20
14. References	21
14.1. Normative References	21
14.2. Informative References	22
Appendix A. Design Considerations	22
A.1. General	22
A.2. UA Interoperability	23
A.3. Usage of port number value zero	24
A.4. B2BUA And Proxy Interoperability	25
A.4.1. Traffic Policing	25
A.4.2. Bandwidth Allocation	25
A.5. Candidate Gathering	26
Authors' Addresses	26

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-mux-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. General

When an Offerer generates an Offer, it assigns an address to each "m=" line, according to the procedures in [RFC3264]. To each "m=" line within a BUNDLE group the Offerer assigns either an address that is unique to that "m=" line, or a shared address that is also assigned to other "m=" lines within the BUNDLE group. Such shared address can be, but does not have to be, a previously selected BUNDLE address Section 6.5.1.

OPEN ISSUE (Q6): There is a discussion on whether assigning a shared address to multiple "m=" lines shall be allowed until the Answerer has indicated support of BUNDLE.

- o (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12245.html>)

The Offerer MUST NOT assign an address (unique or shared), that it has assigned to an "m=" line within a BUNDLE group, to an "m=" line outside the BUNDLE group.

The Offerer MUST, for a BUNDLE group, on the SDP session level [RFC4566], insert an SDP group:BUNDLE attribute associated with the BUNDLE group. The Offerer MUST assign an SDP 'mid' attribute [RFC5888] to each "m=" line within the BUNDLE group, and place the mid value in the group:BUNDLE attribute mid list.

The Offerer MAY assign an SDP 'bundle-only' attribute [ref-to-be-added] to one or more "m=" lines within a BUNDLE group.

OPEN ISSUE (Q8): It still needs to be decided whether a zero port value can be assigned to a 'bundle-only' "m=" line.

- o (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12075.html>)
- o (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12226.html>)
- o (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12339.html>)

6.4.2. Request BUNDLE address selection

When an Offerer generates an Offer, it MUST indicate which address (unique or shared) within a BUNDLE group it wishes the Answerer to select as the Offerer's BUNDLE address for the BUNDLE group Section 6.5.1. The Offerer MUST do this even if the Answerer has, in a previous Answer within the dialog, already selected the Offerer's BUNDLE address.

In order to request an address (unique or shared) to be selected as the Offerer's BUNDLE address for a BUNDLE group, the Offerer places the mid value, associated with the "m=" line representing the requested address, first in the SDP group:BUNDLE attribute mid list associated with the BUNDLE group.

Section 10.1 shows an example of a Bundle Address Request.

6.4.3. Bundle Address Synchronization (BAS)

When an Offerer receives an Answer, in which an offered BUNDLE group is accepted, if the Offerer in the associated Offer assigned an address (unique or shared), that does not represent the BUNDLE address selected for the Offerer, to an "m=" line within the BUNDLE group, the Offerer MUST send a subsequent Offer, in which it assigns the BUNDLE address selected for the Offerer to each "m=" line within the BUNDLE group. This procedure is referred to as Bundle Address Synchronization (BAS), and the Offer is referred to as a BAS Offer.

The Offerer MAY modify any SDP parameter in a BAS Offer.

NOTE: It is important that the BAS Offer 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 BAS Offer. Removing "m=" lines, or reducing the number of codecs, in the BAS Offer used for the is considered to have a low risk of being rejected.

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

Section 10.1 shows an example of an BAS Offer.

6.4.4. Adding a media description to a BUNDLE group

When an Offerer generates an Offer, in which it adds an "m=" line to a BUNDLE group, the Offerer assigns an address (unique or shared) to

the "m=" line, assigns an SDP 'mid' attribute to the "m=" line, and places the mid value in the group:BUNDLE attribute mid list associated with the BUNDLE group, according to the procedures in Section 6.4.2. If the Offerer wishes the Answerer to select the address assigned to the added "m=" as the Offerer's BUNDLE address, the mid value associated with the "m=" line is placed first in the list, according to the procedures in Section 6.4.2.

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

6.4.5. Moving A Media Description Out Of A BUNDLE Group

When an Offerer generates an Offer, in which an "m=" line is moved out of a BUNDLE group, the Offerer MUST assign a unique address to the moved "m=" line. In addition, the Offerer MUST NOT anymore include a mid value, representing the "m=" line, in the SDP group:BUNDLE attribute mid list associated with the BUNDLE group.

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

6.4.6. Disabling A Media Description In A BUNDLE Group

When an Offerer generates an Offer, in which an "m=" line associated with a BUNDLE group is disabled, the Offerer MUST assign an address with a zero port value [RFC4566] to the disabled "m=" line. In addition, the Offerer MUST NOT anymore include a mid value, representing the "m=" line, in the SDP group:BUNDLE attribute mid list associated with the BUNDLE group.

OPEN ISSUE (Q8): It still needs to be decided whether a zero port value can be assigned to a 'bundle-only' "m=" line.

- o (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12075.html>)
- o (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12226.html>)
- o (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12339.html>)

Section 10.5 shows an example of an Offer used to disable an "m=" line in a BUNDLE group.

6.5. SDP Answerer Procedures

6.5.1. Offerer Bundle Address Selection

When an Answerer generates an Answer that contains a BUNDLE group, the Answerer MUST select the Offerer's BUNDLE address. The first mid value in the SDP group:BUNDLE attribute mid list of the Offer represents the address which the Offerer wishes the Answer to select as the Offerer's BUNDLE address Section 6.4.2.

The Answerer SHOULD select the address represented by the first mid value, unless the Answerer in the associated Answer will reject the "m=" line associated with the mid value, or remove the "m=" line from the BUNDLE group. In such case the Answerer MUST select an address associated with the first unrejected mid value that remains in the SDP group:BUNDLE attribute mid list of the Offer.

In the SDP Answer, the Answerer MUST place the mid value associated with the selected Offerer's 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.

6.5.2. Answerer Bundle Address Selection

When an Answerer creates an Answer that contains a BUNDLE group, the Answerer MUST assign a local shared address, the Answerer's BUNDLE address, to each "m=" line within the BUNDLE group.

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

The Answerer MUST NOT assign a shared address, that it has assigned to an "m=" line within a BUNDLE group, to an "m=" line outside the BUNDLE group.

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

6.5.3. Moving A Media Description Out Of A BUNDLE Group

When an Answerer generates an Answer, in which an "m=" line is moved out of a BUNDLE group, the Answerer assigns an address to the moved "m=" line based on the type of address that the Offerer assigned to the associated "m=" line in the associated Offer, as described below.

If the Offerer assigned a shared address to the "m=" line, the answerer MUST reject the moved "m=" line, according to the procedures in Section 6.5.4.

If the Offerer assigned an SDP 'bundle-only' attribute to the "m=" line, the Answerer MUST reject the moved "m=" line, according to the procedures in Section 6.5.4.

If the Offerer assigned a unique address to the "m=" line, the Answerer MUST assign a unique address to the moved "m=" line.

In addition, in either case above, the Answerer MUST NOT anymore include a mid value, representing the "m=" line, in the SDP group:BUNDLE attribute list associated with the BUNDLE group.

6.5.4. Rejecting A Media Description In A BUNDLE Group

When an Answerer generates an Answer, in which an "m=" line associated with a BUNDLE group is rejected, the Answerer MUST assign an address with a zero port value to the rejected "m=" line, according to the procedures in [RFC4566]. In addition, the Answerer MUST NOT anymore include a mid value, representing the "m=" line, in the SDP group:BUNDLE attribute midlist associated with the BUNDLE group.

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 SDES 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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 biloxi.example.com
s=
c=IN IP4 biloxi.example.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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 biloxi.example.com
s=
c=IN IP4 biloxi.example.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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 biloxi.example.com
s=
c=IN IP4 biloxi.example.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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 biloxi.example.com
s=
c=IN IP4 biloxi.example.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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 biloxi.example.com
s=
c=IN IP4 biloxi.example.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-04

- o Updated Offerer procedures (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12293.html>).
- o Updated Answerer procedures (<http://www.ietf.org/mail-archive/web/mmusic/current/msg12333.html>).
- o Usage of SDP 'bundle-only' attribute added.

- o Reference to Trickle ICE document added.

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-mux-attributes]
Nandakumar, S. and C. Jennings, "A Framework for SDP Attributes when Multiplexing ", draft-nandakumar-mmusic-sdp-mux-attributes-04 (work in progress), September 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.

[I-D.ietf-mmusic-trickle-ice]
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol ", draft-ietf-mmusic-trickle-ice-00 (work in progress), October 2013.

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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 biloxi.example.com
s=
c=IN IP4 biloxi.example.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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 B2BUS 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 atlanta.example.com
s=
c=IN IP4 atlanta.example.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 [I-D.ietf-mmusic-trickle-ice] 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

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.

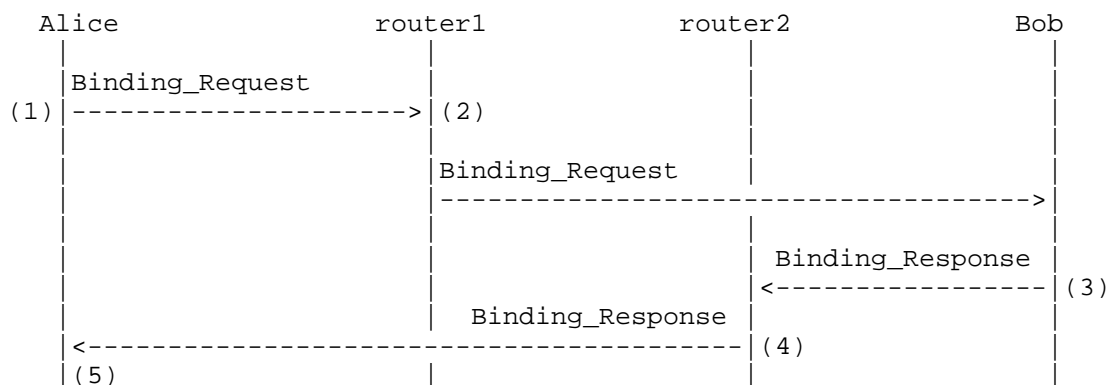
The MALICE Server agent MAY also include the new MD-PEER-CHECK-RES TLV defined in this specification if it has already performed a MALICE check and has the result available. This is useful if the MALICE Server is the controlled agent and wishes to influence the nomination process at MALICE Client (controlling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

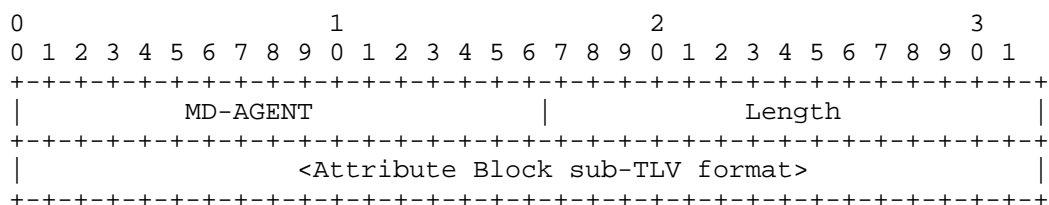


Figure 1: MD-AGENT Attribute

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.

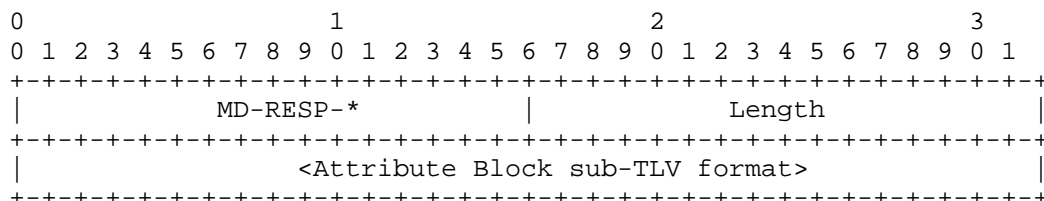


Figure 2: MD-RESP- Attribute

Where MD-RESP-* = {MD-RESP-UP | MD-RESP-DN}

9.1.3. MD-PEER-CHECK Attribute

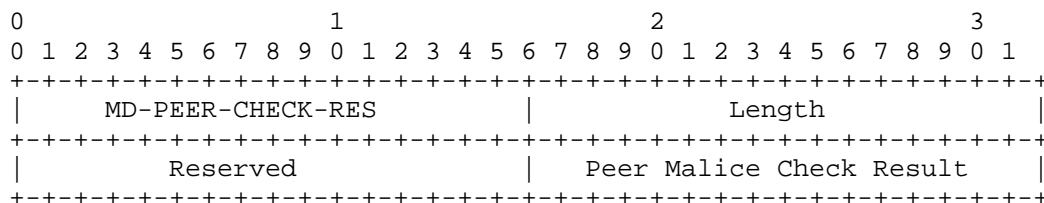


Figure 3: MD-PEER-CHECK Attribute

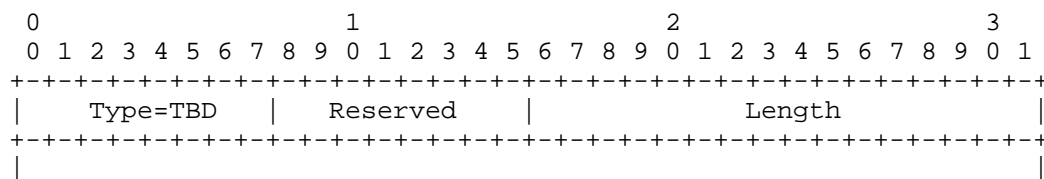
Peer Malice Check Result - "Success" or "Failure".

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.



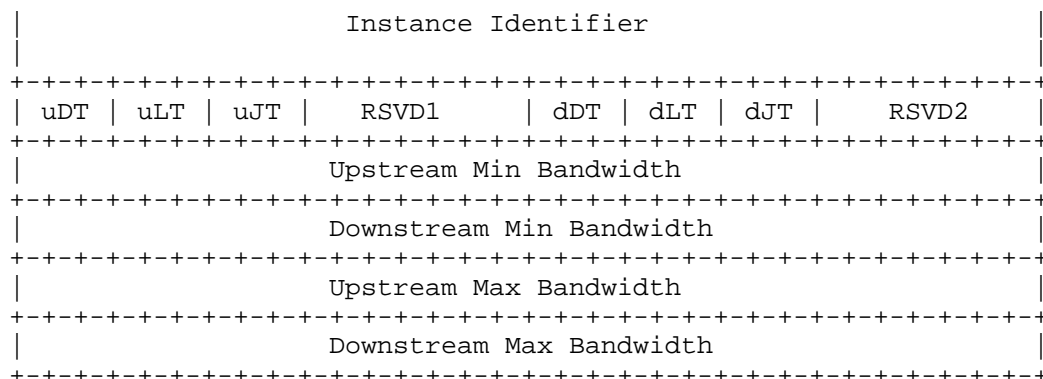


Figure 4: FLOWDATA Request

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.

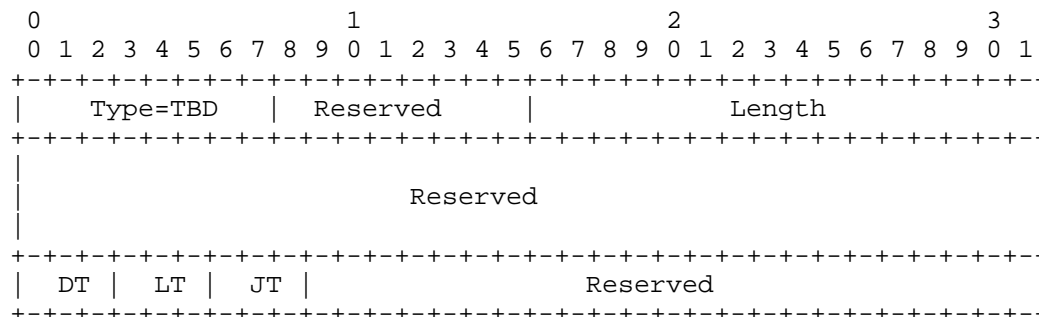
Upstream Maximum Bandwidth: Maximum Upstream bandwidth in bytes per second, 0 means no information is available.

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.



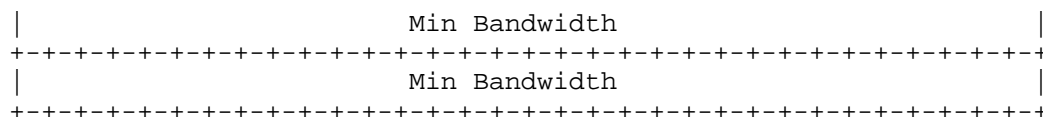


Figure 5: FLOWDATA Response

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2014

S. Nandakumar
Cisco
October 18, 2013

A Framework for SDP Attributes when Multiplexing
draft-nandakumar-mmusic-sdp-mux-attributes-05

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 April 21, 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	9
5.1. RFC4566 - SDP: Session Description Protocol	10
5.2. RFC4585 - RTP/AVPF	11
5.3. RFC5761 - Multiplexing RTP and RTCP	12
5.4. RFC4574 - SDP Label Attribute	13
5.5. RFC5432 - QoS Mechanism Selection in SDP	13
5.6. RFC4568 - SDP Security Descriptions	14
5.7. RFC5762 - RTP over DCCP	14
5.8. RFC6773 - DCCP-UDP Encapsulation	15
5.9. RFC5506 - Reduced-Size RTCP in RTP Profile	16
5.10. RFC6787 - Media Resource Control Protocol Version 2	16
5.11. RFC5245 - Interactive Connectivity Establishment (ICE)	17
5.12. RFC5285 - RTP Header Extensions	17
5.13. RFC3605 - RTCP attribute in SDP	18
5.14. RFC5576 - Source-Specific SDP Attributes	18
5.15. RFC6236 - Image Attributes in SDP	19
5.16. RFC6285 - Rapid Acquisition of Multicast RTP Sessions	20
5.17. RFC6230 - Media Control Channel Framework	20
5.18. RFC6364 - SDP Elements for FEC Framework	21
5.19. RFC4796 - Content Attribute	21
5.20. RFC3407 - SDP Simple Capability Declaration	22
5.21. RFC6284 - Port Mapping between Unicast and Multicast RTP Sessions	22
5.22. RFC6714 - MSRP-CEMA	23
5.23. RFC4583 - SDP Format for BFCP Streams	23
5.24. RFC5547 - SDP Offer/Answer for File Transfer	24
5.25. draft-ietf-mmusic-media-loopback	24
5.26. RFC5760 - RTCP with Unicast Feedback	25
5.27. RFC3611 - RTCP XR	25
5.28. RFC5939 - SDP Capability Negotiation	25
5.29. draft-ietf-mmusic-sdp-media-capabilities	26

5.30.	RFC4567 - Key Management Extensions for SDP and RTSP . . .	27
5.31.	RFC4572 - Comedia over TLS in SDP	27
5.32.	RFC4570 - SDP Source Filters	28
5.33.	RFC6128 - RTCP Port for Multicast Sessions	28
5.34.	RFC6189 - ZRTP	29
5.35.	RFC4145 - Connection-Oriented Media	30
5.36.	RFC5159 - OMA BCAST SDP Attributes	30
5.37.	RFC6193 - Media Description for IKE in SDP	31
5.38.	RFC6064 - SDP and RTSP Extensions for 3GPP	32
5.39.	RFC3108 - ATM SDP	34
5.40.	3GPP TS 24.182	35
5.41.	3GPP TS 24.183	36
5.42.	3GPP TS 24.229	36
5.43.	ITU T.38	37
5.44.	ITU-T H.248.15	37
5.45.	RFC4975 - The Message Session Relay Protocol	38
5.46.	Historical	39
6.	bwtype Attribute Analysis	39
6.1.	RFC4566 - SDP: Session Description Protocol	39
6.2.	RFC3556 - SDP Bandwidth Modifiers for RTCP Bandwidth . . .	40
6.3.	RFC3890 - Bandwidth Modifier for SDP	41
7.	rtcp-fb Attribute Analysis	41
7.1.	RFC4585 - RTP/AVPF	41
7.2.	RFC5104 - Codec Control Messages in AVPF	42
7.3.	RFC6285 - Unicast-Based RAMS	43
7.4.	RFC6679 - ECN for RTP over UDP/IP	43
7.5.	RFC6642 - Third-Party Loss Report	43
7.6.	RFC5104 - Codec Control Messages in AVPF	44
8.	group Attribute Analysis	44
8.1.	RFC5888 - SDP Grouping Framework	44
8.2.	RFC3524 - Mapping Media Streams to Resource Reservation Flows	45
8.3.	RFC4091 - ANAT Semantics	45
8.4.	RFC5956 - FEC Grouping Semantics in SDP	46
8.5.	RFC5583 - Signaling Media Decoding Dependency in SDP . . .	46
9.	ssrc-group Attribute Analysis	47
9.1.	RFC5576 - Source-Specific SDP Attributes	47
10.	QoS Mechanism Token Analysis	47
10.1.	RFC5432 - QoS Mechanism Selection in SDP	47
11.	k= Attribute Analysis	47
11.1.	RFC4566 SDP: Session Description Protocol	48
12.	content Attribute Analysis	48
12.1.	RFC4796 - MSRP Relays	48
13.	Payload Formats	48
13.1.	RFC5109 - RTP Payload Format for Generic FEC	48
14.	IANA Considerations	49
15.	Security Considerations	50
16.	Acknowledgments	50

17. Change Log	50
18. References	51
18.1. Normative References	51
18.2. Informative References	51
Author's Address	57

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=" line 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 video media line's a=crypto attribute is chosen since its mid value (bar) appears first in the a=group:BUNDLE line. This is due to BUNDLE grouping semantic [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 bar foo
m=audio 49172 RTP/AVP 99
a=mid:foo
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:bar
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 "Level" 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	Level	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	Level	Category
rtcp-fb	RTCP reporting happens per RTP Session.	M	IDENTICAL

RFC4585 Attribute Analysis

OPEN ISSUE: Below SDP examples show 3 scenarios on implications of allowing Payload Types to be repeated or not repeated within the BUNDLED m= lines.

Scenario1: Payload Types are not shared and rtcp-fb is mandated for all the multiplexed m= lines. This scenario leads into assigning category "NOT IMPACTED" for rtcp-fb and its parameters since RTCP reporting happens per PayloadType within a BUNDLED RTP Session, thus allowing for unique reporting.

```
a=group:BUNDLE audio video
m=audio 3456 RTP/AVP 97
a=mid:audio
a=rtpmap:97 iLBC/8000
m=video 3456 RTP/AVP 98
a=mid:video
a=rtpmap:98 VP8/90000
a-rtcp-fb:98 nack rpsi
```

Scenario2: Payload Types are shared and rtcp-fb is repeated across all the multiplexed m= lines with same feedback type (nack). This leads to assigning category "IDENTICAL" to rtcp-fb which allows NACK and NACK RPSI reporting to be done for PT 98 in this example.

```

a=group:BUNDLE audio video
m=audio 3456 RTP/AVP 98
a=mid:audio
a=rtpmap:98 iLBC/8000
a=rtcp-fb:98 nack
m=video 3456 RTP/AVP 98
a=mid:video
a=rtpmap:98 VP8/90000
a-rtcp-fb:98 nack rpsi

```

In the below case, audio media line has rtcp-fb ack and video media line has rtcp-fb nack with Payload types shared. With such a configuration, positive acks are reported for video stream even it was not intended and negative acks are reported for audio stream unintended.

```

// PTs are shared and have different feedback types
a=group:BUNDLE audio video
m=audio 3456 RTP/AVP 98
a=mid:audio
a=rtpmap:98 iLBC/8000
a=rtcp-fb ack // Positive ACK
m=video 3456 RTP/AVP 98
a=mid:video
a=rtpmap:98 VP8/90000
a-rtcp-fb:98 nack rpsi // Nack ACK

```

OPEN ISSUE: Should BUNDLE disallow PT to be repeated ?? What does it mean if PTs are shared in the context of rtcp-fb ?? Magnus pointed out that RTCP FB reporting MUST happen per (PT+Media Source) when BUNDLED. If this is allowed, the category of "IDENTICAL" applies with rtcp-fb for all m= lines and update BUNDLE spec to mandate the behavior. Does this sound fine ?

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	Level	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	Level	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	Level	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	Level	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	Level	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	Level	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	Level	Category
resource	Not Impacted	M	NORMAL
channel	Not Impacted	M	NORMAL
	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	Level	Category
ice-lite	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	M	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	Level	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	Level	Category
rtcp	Case1:Same RTCP port is repeated across the m=lines. Case2:Different RTCP ports renders multiplexing impossible	M	TRANSPORT

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	Level	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 9 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

OPEN ISSUE:LNx: There has been concern on how to deal with SSRC values repeated across the multiplexed m= lines. Lennox suggested to possible options to deal with it. The simpler is to say that individual SSRC values in a=ssrc attributes MUST NOT be repeated across bundled m-lines, but other than that the behavior is normal. The more complicated is to say that SSRC attributes MAY be repeated across bundled m-lines, iff the media stream satisfies more than one m-line. In this case, it's up to the individual source attributes to define what's allowed.

Should BUNDLE allow SSRCs to be repeated ? If so, should the category for the source attributes be SPECIAL and update RFC5576 to add text to define multiplexing scenarios.

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	Level	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	Level	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	Level	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	Level	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	Level	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	Level	Category
sqn	Not Impacted	B	NORMAL
csdc	Mismatch in the offered capability description MAY fail multiplexing.	B	TBD
cpar	Mismatch in the offered capability parameters MAY fail multiplexing.	B	TBD
cparmin	Mismatch in the offered capability parameters MAY fail multiplexing.	B	TBD
cparmax	Mismatch in the offered capability parameters MAY fail multiplexing.	B	TBD

RFC3407 Attribute 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	Level	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	Level	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	Level	Category
floorctrl	Must be repeated across all the multiplexed m=lines	M	IDENTICAL
confid	Not Impacted	M	NORMAL
userid	Not Impacted	M	NORMAL
floorid	The floorid MUST be unique across the multiplexed m=lines	M	NOT RECOMMENDED

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	Level	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	Level	Category
loopback rtp-pkt-loopback	Not Impacted	M	NORMAL
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	Level	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	Level	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	Level	Category
pcfg	Depends on capability being negotiated	M	SPECIAL
acfg	Depends on capability being negotiated	M	SPECIAL
csup	Depends on capability being negotiated	B	SPECIAL
creq	Depends on capability being negotiateds	B	SPECIAL
acap	Depends on capability being negotiated	B	SPECIAL
tcap	Repeat transport capability across all m= lines	B	IDENTICAL

RFC5939 Attribute Analysis

5.29. draft-ietf-mmusic-sdp-media-capabilities

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	Level	Category
rmcap	Not Impacted	B	NORMAL
omcap	Not Impacted	B	NORMAL
mfcap	Not Impacted	B	NORMAL
mscap	Not Impacted	B	NORMAL
lcfg	Not Impacted	B	NORMAL
secap	Not Impacted	S	NORMAL

draft-ietf-mmusic-sdp-media-capabilities Attribute 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	Level	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	Level	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	Level	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	Level	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	Level	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	Level	Category
setup	Should be identical across all m=lines	B	IDENTICAL
connection	Should be identical across all m=lines	B	IDENTICAL

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	Level	Category
bcastversion		S	TBD
stkmstream		B	TBD
SRTPAuthentication		M	TBD
SRTPROCTxRate		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	Level	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	Level	Category
X-predecbufsize	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing with audio m=lines is invalid	M	NOT RECOMMENDED
X-initpredecbufperiod	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing with audio m=lines is invalid	M	NOT RECOMMENDED
X-initpostdecbufperiod	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing with audio m=lines is invalid	M	NOT RECOMMENDED
X-decbyterate	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing 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 mulitple m=lines of different types cannot be clearly specified	M	TBD
alt-group	Complicates selection of alternate m=lines grouped with alt-group on mulitplexing	M	TBD
3GPP-Adaptation-Support		M	TBD
3GPP-QoE-Metricsn		B	TBD
3GPP-Asset-Informatio		B	TBD
mbms-mode		B	TBD
mbms-flowid	Multiplexing multiple m=lines complicates FEC mappings to the transport addresses.	M	TBD
mbms-repair		B	TBD

+-----	+-----	+-----	+-----	+-----

RFC6064 Attribute Analysis

OPEN ISSUE:MW: These paramters are defined for the declarative usage in RTSP or multicast/broadcast and dont have O/A defintions. Does BUNDLE apply only to O/A usage ?

5.39. RFC3108 - ATM SDP

RFC3108 [RFC3108] describes conventions for using the Session Description Protocol (SDP) described for controlling ATM Bearer Connections, and any associated ATM Adaptation Layer (AAL)

Name	Notes	Level	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

aal5sscop	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
dset	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

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.

5.40. 3GPP TS 24.182

3GPP TS 24.182 [3GPP TS 24.182] specifies IP multimedia subsystem Custom Alerting tones

Name	Notes	Level	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	Level	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	Level	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	Level	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	Level	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	Level	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	Level	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	Level	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	Level	Category
bwtype:RS	S level usage represents session aggregate and media level usage indicates SUM of the individual values while multiplexing	B	NORMAL, SUM
bwtype:RR	S 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	Level	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	Level	Category
ack rpsi	Not Impacted	M	NORMAL
ack app	Feedback parameters MUST be handled in the app specific way when multiplexed	M	SPECIAL
nack	Not Impacted	M	NORMAL
nack pli	Not Impacted	M	NORMAL
nack sli	Not Impacted	M	NORMAL
nack rpsi	Not Impacted	M	NORMAL
nack app	Feedback parameters MUST be handled in the app specific way when multiplexed	M	SPECIAL
trr-int	This attribute applies to RTP Session as a whole	M	IDENTICAL

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	Level	Category
ccm	Not Impacted	M	Normal

RFC5104 Attribute Analysis

7.3. RFC6285 - Unicast-Based RAMS

Name	Notes	Level	Category
nack rai	Not Impacted	M	NORMAL

RFC6285 Attribute Analysis

7.4. 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	Level	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

7.5. 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	Level	Category
nack tllei	Not Impacted	M	NORMAL
nack pslei	Not Impacted	M	NORMAL

RFC6642 Attribute Analysis

7.6. RFC5104 - Codec Control Messages in AVPF

Attr Name	Notes	Level	Category
ccm fir	Not Impacted	M	NORMAL
ccm tmnbr	Not Impacted	M	NORMAL
ccm tstr	Not Impacted	M	NORMAL
ccm vbcm	Not Impacted	M	NORMAL

RFC5104 Attribute Analysis

8. group Attribute Analysis

This section analyzes SDP "group" semantics [GROUP-SEM].

8.1. RFC5888 - SDP Grouping Framework

RFC5888 [RFC5888] defines a framework to group "m" lines in the Session Description Protocol (SDP) for different purposes.

Name	Notes	Level	Category
group:LS	Not Impacted	S	NORMAL
group:FID	Not Impacted	S	NORMAL

RFC5888 Attribute Analysis

8.2. 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	Level	Category
group:SRF	Not Impacted	S	NORMAL

RFC3524 Attribute Analysis

8.3. 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	Level	Category
group:ANAT	Not Impacted	S	NOT RECOMMENDED

RFC4091 Attribute Analysis

8.4. 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	Level	Category
group:FEC-FR	Not Impacted	S	NORMAL

RFC5956 Attribute Analysis

8.5. 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	Level	Category
depend lay	Not Impacted	M	NORMAL
depend mdc	Not Impacted	M	NORMAL

RFC5583 Attribute Analysis

OPEN ISSUE:Eckel: If PTs are not unique across the m=lines BUNDLED, how should this be handled ?

9. ssrc-group Attribute Analysis

This section analyzes "ssrc-group" semantics [SSRC-GROUP].

9.1. RFC5576 - Source-Specific SDP Attributes

Name	Notes	Level	Category
FID	Not Impacted	M	NORMAL
FEC	Not Impacted	M	NORMAL
FEC-FR	Not Impacted	M	NORMAL

RFC5576 Attribute Analysis

10. QoS Mechanism Token Analysis

This section analyzes QoS tokens specified with SDP[QOS].

10.1. RFC5432 - QoS Mechanism Selection in SDP

Name	Notes	Level	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

11. k= Attribute Analysis

11.1. RFC4566 SDP: Session Description Protocol

Name	Notes	Level	Category
k=	It is NOT recommended to use this attribute	S	NOT RECOMMENDED

RFC4566 Attribute Analysis

12. content Attribute Analysis

12.1. RFC4796 - MSRP Relays

Name	Notes	Level	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

13. Payload Formats

13.1. 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	Level	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.

14. 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

15. Security Considerations

All the attributes which involve security key needs a careful review to ensure two-time pad vulnerability is not created

16. Acknowledgments

I would like to thank Cullen Jennings for suggesting the categories, contributing text and helping review the draft. I would also link to thank Magnus, Christer and Dan on suggesting structural changes helping improve the document readability.

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 (5.36,5.42,5.43,5.44), Colin Perkins(5.7,5.8), Magnus westerlund(5.3,5.9,6.1,6.2,6.3,8.3,5.2,7,8,9), Roni Evens(5.12,5.27,8.4), Subha Dhesikan(5.5,12.1), Dan Wing(5.6,5.11,5.30,5.34,5.37), Ali C Begen(5.1,5.16,5.18,5.21,5.33,8.2,13.1,10.4,15.1), Bo Burman (7.2,7.6), Charles Eckel(5.14,5.23,5.24,10.5), Paul Kyzivat(5.24), Ian Johansson (5.11), Saravanan Shanmugham(5.10), Paul E Jones(5.25), Rajesh Kumar (5.39), Jonathan Lennox (5.31,5.14,11.1), Mo Zanaty(5.4,5.19,10.1,10.5)

17. Change Log

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

Changes from draft-nandakumar-mmusic-mux-attributes-04

- o Added few OPEN ISSUES that needs to be discussed.
- o Updated sections 5.10,5.23,5.24,5.25,7.2,9.1,5.12,5.27,8.4, 5.44,5.11,5.4,5.19,10.1,10.5,5.21,10.4,15.1
- o Updated Table Column name Current to Level and improved TRANSPORT category explanation on suggestions from Dan Wing.
- o Grouped all the rtcp-fb attribute analysis under a single section as suggested by Magnus/

Changes from draft-nandakumar-mmusic-mux-attributes-03

- o Maintenance change to clean up grammatical nits and wordings.

Changes from draft-nandakumar-mmusic-mux-attributes-02

- o Updated Sections 5.3,5.5,5.6,5.7,5.9,5.8,5.11,5.13,5.22,5.34, 5.37,5.40,5.41,5.42,5.43,5.44,5.45,6.1,6.2,6.3,8.3,12.1 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.

18. References

18.1. 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.

18.2. 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]
"S Description Protocol (SDP) RTCP ACK/NACK Feedback attributes", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-15>>.
- [CCM]
"S Description Protocol (SDP) RTCP-FB Codec Control Messages", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-19>>.
- [GROUP-SEM]
"S Description Protocol (SDP) "group" semantics", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-18>>.

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]

"S Description Protocol (SDP) Parameters", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml>>.

[MEDIA_CAP]

Kaplan, H., Hedayat, K., and N. Venna, "S 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]

"S 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] Andreassen, F., "S 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., "S 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] Andreassen, F., Baugher, M., and D. Wing, "S Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4570] Quinn, B. and R. Finlayson, "S 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., "S 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, "S 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., "S 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., "S 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] "S Description Protocol (SDP) RTCP Feedback attributes", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-14>>.
 - [SSRC-GROUP]
"S 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

MMUSIC
Internet-Draft
Intended status: Informational
Expires: April 13, 2014

T. Reddy
D. Wing
B. VerSteeg
R. Penno
Cisco
V. Singh
Aalto University
October 10, 2013

Improving ICE Interface Selection Using Port Control Protocol (PCP) Flow
Extension
draft-reddy-mmusic-ice-best-interface-pcp-00

Abstract

A host with multiple interfaces needs to choose the best interface for communication. Oftentimes, this decision is based on a static configuration and does not consider the link characteristics of that interface, which may affect the user experience.

This document describes a mechanism for an endpoint to query the link characteristics from the access router (the router at the other end of the endpoint's access link) using a Port Control Protocol (PCP) Flow Extension. This information influences endpoint's Interactive Connectivity Establishment (ICE) candidate selection algorithm.

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 April 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. Notational Conventions	3
3. Algorithm overview	3
3.1. Changed Link Quality	4
4. Multiple Interfaces	5
4.1. Multiple Interfaces for media streams	5
4.2. Availability of New Interfaces	5
5. IANA Considerations	6
6. Security Considerations	6
7. Acknowledgements	6
8. References	6
8.1. Normative References	6
8.2. Informative References	6
Appendix A.	7
A.1. Delay Factor in Discovering the Link Characteristics	7
Authors' Addresses	7

1. Introduction

ICE [RFC5245] uses a prioritization formula to perform connectivity checks, in which the most preferred address pairs are tested first and when a sufficiently good pair is discovered, the ICE connectivity tests are stopped. ICE prefers address pairs in the following order: transport address directly attached to the endpoint's network interface (host candidate), transport address on the public side of a NAT (server reflexive candidate), and finally, transport address that are allocated on a media relay (relayed candidate). This approach works well for an endpoint with a single interface, but is too simplistic for endpoints with multiple interfaces. The network interfaces may have different link characteristics, but that will not be known without the awareness of the upstream and downstream characteristics of the access link.

In this document, an ICE agent [RFC5245] uses PCP Flow Extension [I-D.wing-pcp-flowdata] to determine the link characteristics of the host's interfaces, which influence the ICE candidate priority.

As this document explains the interworking of ICE and Port Control Protocol (PCP) Flow Extensions, it is beneficial to first read the Overview of ICE (Section 2 of ICE [RFC5245]) and PCP Flowdata Option [I-D.wing-pcp-flowdata]. Additionally, PCP for WebRTC [I-D.penno-rtcweb-pcp] describes the problems with traversing NATs and firewalls, current techniques used to solve them and the PCP solution in these scenarios.

2. Notational Conventions

This note uses terminology defined in ICE [RFC5245] and PCP [RFC6887].

3. Algorithm overview

The proposed algorithm is backward compatible with existing implementations, and does not require any changes other than to the selection of candidate priority.

When an endpoint first joins a network, it determines if the network supports PCP Flow Extensions by following the procedures described in [I-D.wing-pcp-flowdata]. Basically, the endpoint sends a PCP Flow Extension probe packet, the response to which provides coarse information on the link capabilities. After confirming that PCP Flow Extensions are supported on that network interface, the ICE agent can use PCP Flow Extensions on that interface (rather than STUN).

When a media session needs to be established, and the user and operator controlled policies on an endpoint permit more than one interface for a media session, the ICE agent uses PCP Flow Extensions to (a) obtain a mapping from its NAT or firewall and (b) determine the characteristics of the link. After receiving the PCP Flow Extension responses from its various interfaces, the ICE agent sorts the ICE candidates according to the link capacity characteristics. ICE candidates from the interface which best fulfills the desired flow characteristics is assigned the highest priority and the best suited interface should be used to communicate with the TURN server to learn the relayed candidate address.

The ICE agent calculates the priorities of host and server-reflexive candidates based on the above steps and signals these candidates in offer or answer to the remote peer. After the offer and answer are exchanged, the participating ICE agents begin pairing the candidates, ordering them into check lists to start the ICE connectivity check phase and eventually select the pair of candidates that will be used for real-time communication.

3.1. Changed Link Quality

It is possible that the characteristics of a link may change over time, and therefore the ICE agent may want to move the media to a different interface. For example, if a competing high-bandwidth flow starts or finishes its data transmission; the DSL line rate might have improved (or degraded); the link capacity may have been dynamically increased (or decreased). When link quality changes in such a fashion, the PCP Flow Extensions sends a PCP message to the endpoint. Upon receiving the message, the ICE agent may decide to move the active flow to a more suitable interface and performs ICE restart to trigger the switch over of the media streams to the new interface.

For ICE local relayed candidates, the ICE agent can switch to the more suitable interface by refreshing its allocation with the TURN server using the procedures explained in section 5 of Mobility using TURN [I-D.wing-mmusic-ice-mobility]. Thus reusing the local relayed candidate on a different interface even if the endpoint IP address changes. Therefore, the ICE agent can switch over local relayed candidate to the most suited interface that meets the requirements of the media stream. This way, even without informing the SIP server and remote peer, ICE agent can switch over a local relayed candidate to the most suited interface which meets the requested flow characteristics.

4. Multiple Interfaces

If multiple interfaces are available, the ICE agent can use PCP Flow Extensions [I-D.wing-pcp-flowdata] to determine the best path. The advantage is PCP can be used to select the most suitable interface for the media streams. When an endpoint has multiple interfaces (for example 3G, 4G, WiFi, VPN, etc.), an ICE agent can choose the interfaces for media streams according to the path characteristics, as discussed in the previous section.

4.1. Multiple Interfaces for media streams

If the requested flow characteristics for the media streams cannot be handled by a single interface but by multiple interfaces then the ICE agent performs the following steps:

- o ICE agent based on the ICE connectivity results could select multiple interfaces for the media session. For example, the ICE agent selects to send the audio stream over the WiFi access point because it offers (via PCP Flow Extensions) low delay, low packet loss and average capacity of 120 Kbps, but for the video stream it selects the 3G interface because it offers medium delay, medium packet loss and average capacity of 500Kbps.
- o Alternatively, the ICE agent on a mobile device may also want to select the best suited interface among all the available interfaces even if it does not serve the requested flow characteristics for all the media streams, so that other interfaces can be turned off to increase the battery life of cellular connected devices such as smartphones or tablets.

4.2. Availability of New Interfaces

If the available interfaces do not meet the requested flow characteristics then ICE agent can either proceed as usual using the "Recommended Formula" explained in Section 4.1.2.1 of [RFC5245] to prioritize the candidates or use the Happy Eyeballs Extension for ICE algorithm proposed in [I-D.reddy-mmusic-ice-happy-eyeballs] for dual-stack endpoint. When new interfaces become available then ICE agent can use PCP Flow Extension to find if the newly available interfaces meet the flow characteristics. When a PCP response is received from at least one of the new interfaces and if it meets the requirements, the endpoint can re-connect to the SIP proxy using the new interface. The endpoint uses the candidates indicated in the previous PCP response, it exchanges updated offer/answer to trigger ICE restart. Once the ICE processing reaches the "Completed state", the ICE endpoint can successfully switch the media session over to the new interface. The interface initially used for communication can now be

turned off without disrupting communications.

5. IANA Considerations

None.

6. Security Considerations

Security considerations discussed in [RFC6887] are to be taken into account.

7. Acknowledgements

Authors would like to thank Anca Zamfir for comments and review.

8. References

8.1. Normative References

- [I-D.wing-pcp-flowdata]
Wing, D., Penno, R., and T. Reddy, "PCP Flowdata Option",
draft-wing-pcp-flowdata-00 (work in progress), July 2013.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment
(ICE): A Protocol for Network Address Translator (NAT)
Traversal for Offer/Answer Protocols", RFC 5245,
April 2010.
- [RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P.
Selkirk, "Port Control Protocol (PCP)", RFC 6887,
April 2013.

8.2. Informative References

- [I-D.penno-rtcweb-pcp]
Penno, R., Reddy, T., Wing, D., and M. Boucadair, "PCP
Considerations for WebRTC Usage",
draft-penno-rtcweb-pcp-00 (work in progress), May 2013.
- [I-D.reddy-mmusic-ice-happy-eyeballs]
Reddy, T., Patil, P., and D. Wing, "Happy Eyeballs
Extension for ICE",
draft-reddy-mmusic-ice-happy-eyeballs-03 (work in
progress), October 2013.

[I-D.wing-mmusic-ice-mobility]

Wing, D., Reddy, T., Patil, P., and P. Martinsen,
"Mobility with ICE (MICE)",
draft-wing-mmusic-ice-mobility-05 (work in progress),
September 2013.

Appendix A.

A.1. Delay Factor in Discovering the Link Characteristics

Some concern has been expressed, that discovering the link characteristics may consume more time than using STUN. However, STUN will actually take more time than learning link characteristics, because a STUN request/response traverses across more routers than a PCP Flow Extension request.

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

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Bill VerSteeg
Cisco Systems, Inc.
5030 Sugarloaf Parkway
Lawrenceville 30044
USA

Email: billvs@cisco.com

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, 95134
USA

Phone:
Email: repenno@cisco.com
URI:

Varun Singh
Aalto University
School of Electrical Engineering
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: varun.singh@iki.fi
URI: <http://www.netlab.tkk.fi/~varun/>

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: April 11, 2014

T. Reddy
P. Patil
D. Wing
Cisco
October 08, 2013

Happy Eyeballs Extension for ICE
draft-reddy-mmusic-ice-happy-eyeballs-03

Abstract

This document describes an algorithm that makes Interactive Connectivity Establishment (ICE) connectivity checks more responsive by reducing delays in dual-stack host ICE connectivity checks when there is a path failure for an address family preferred by the application or by the operating system. As IPv6 is usually preferred over IPv4, 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 April 11, 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. Candidate Priority	3
4. Algorithm overview	3
4.1. Processing the Results	4
5. IANA Considerations	5
6. Security Considerations	5
7. Acknowledgements	5
8. References	6
8.1. Normative References	6
8.2. Informative References	6
Authors' Addresses	7

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 causes ICE to perform terribly in cases where IPv6 doesn't work, which is still very commonplace. This document recommends an alternative prioritization for candidates that improves this situation with a goal that the ICE agent not be inordinately harmed by a simple reordering of the candidates.

This document describes an algorithm that makes ICE connectivity checks more responsive to failures of an address family by reordering the candidates such that IPv6 and IPv4 candidates get a fair chance during connectivity checks. This algorithm change is backward compatible with existing implementations, and does not require any changes other than to the selection of candidate priority.

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. Candidate Priority

By using the technique described in Section 4, if there are both IPv6 and IPv4 addresses candidates gathered, and the first 'N' candidates are of the same IP address family, then the highest-priority candidate of the other address family is promoted to position 'N+1' in the check list thus making ICE connectivity checks more responsive to failures of an address family. The algorithm ensures that there are no more than a fixed number of candidates of a given IP version in a single sequence.

Even if an administrator changes the policy table to prefer IPv4 addresses over IPv6 addresses as explained in [RFC6724], the IPv4 server-reflexive candidates will still have lower priority than IPv6 host candidates as per the "Recommended Formula" (section 4.1.2.1 of [RFC5245]) which is not desired. The Happy Eyeballs extension for ICE algorithm resolves the problem in this scenario as well by ensuring that IPv4 server-reflexive candidates are placed before IPv6 host candidates and thus ordering based on candidate types is no longer in effect.

4. Algorithm overview

The Happy Eyeballs Extension for ICE algorithm proposes the following steps after candidates are prioritized using the formula in section 4.1.2.1 of [RFC5245]:

- a. If the first 'N' candidates are of the same IP address family, then the highest-priority candidate of the other address family is promoted to position 'N+1' in the list.
- b. Step (a) is repeated for subsequent candidates in the list until all candidates of the preferred address family are exhausted.

The algorithm ensures that a long sequence of candidates belonging to the same address family is interleaved with candidates from an alternative IP version.

The following figure illustrates the result of the algorithm on candidates:

Before Happy Eyeballs Extension for ICE algorithm :

```
-----  
(highest)  IPv6 Host Candidate-1  
           IPv6 Host Candidate-2  
           IPv6 Host Candidate-3  
           IPv6 Host Candidate-4  
           IPv6 Host Candidate-5
```

```

IPv6 Host Candidate-6
IPv6 Host Candidate-7
IPv4 Host Candidate
IPv6 Server Reflexive Candidate
IPv4 Server Reflexive Candidate
IPv6 Relayed Transport Candidate
(lowest) IPv4 Relayed Transport Candidate

```

After Happy Eyeballs Extension for ICE algorithm :

```

-----
(highest) IPv6 Host Candidate-1
          IPv6 Host Candidate-2
          IPv6 Host Candidate-3
          IPv4 Host Candidate          ---> Promoted candidate
          IPv6 Host Candidate-4
          IPv6 Host Candidate-5
          IPv6 Host Candidate-6
          IPv4 Server Reflexive Candidate ---> Promoted candidate
          IPv6 Host Candidate-7
          IPv6 Server Reflexive Candidate
          IPv6 Relayed Transport Candidate
(lowest) IPv4 Relayed Transport Candidate

```

4.1. Processing the Results

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 optimizations are 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.



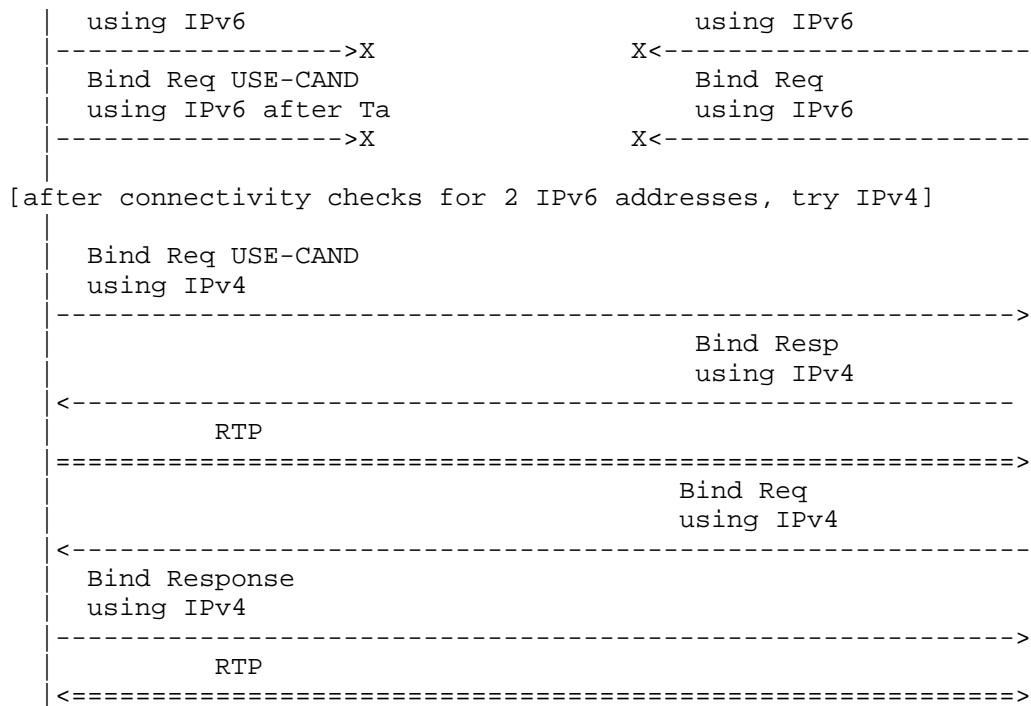


Figure 1: Happy Eyeballs Extension for ICE

5. IANA Considerations

None.

6. 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.

7. Acknowledgements

Authors would like to thank Bernard Aboba, Martin Thomson, Jonathan Lennox, Pal Martinson for their comments and review.

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.
- [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.

8.2. 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.
- [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.
Bangalore
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
Updates: 3264 (if approved)
Intended status: Standards Track
Expires: April 20, 2014

A. B. Roach
Mozilla
S. Nandakumar
Cisco
October 17, 2013

Using Partial Offers and Partial Answers in a Multimedia Session
draft-roach-mmusic-pof-pan-01

Abstract

Whenever two hosts have the ability to set up and control a session on a peer-to-peer basis, situations can arise in which both parties attempt to change session parameters "at the same time," such that the session control messages cross on the wire. When this happens, implementations need to invoke extraordinary procedures to return the shared state of the session to a common view between the endpoints.

For real-time communications, these session control messages are typically exchanged using the session description protocol (SDP), using an Offer/Answer model. This document expands the offer/answer model to include the ability to exchange information relating to discrete media streams within the session. By reducing the amount of session data, the frequency of session state conflicts can be reduced; and, for certain types of operations, conflicts can be eliminated altogether.

This document updates RFC 3264.

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 April 20, 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	4
3. Mechanism Overview	4
3.1. Adding a Stream	5
3.2. Changing a Stream	5
3.3. Removing a Stream	6
4. Use With Other Protocols	6
4.1. High-Level Sketch: Use With JSEP/WebRTC	7
4.2. High-Level Sketch: Use With SIP	7
5. Protocol Operation	7
5.1. Common Procedures	8
5.2. Generating a Partial Offer	8
5.3. Processing a Partial Offer	9
5.4. Processing a Partial Answer	11
5.5. Updating the Shared View of Session State	12
5.6. Receiving a Full Offer with a Partial Offer Pending	12
6. Examples	13
6.1. Adding Streams	14
6.1.1. Full Offer/Answer Procedures	14
6.1.2. Partial Offer/Answer Procedures	15
6.2. Removing Streams	18
6.2.1. Full Offer/Answer Procedures	18
6.2.2. Partial Offer/Answer Procedures	19
6.3. Changing a Stream	21
6.3.1. Full Offer/Answer Procedures	21
6.3.2. Partial Offer/Answer Procedures	22
6.4. Both Sides Simultaneously Add Streams	24
6.4.1. Full Offer/Answer Procedures	24
6.4.2. Partial Offer/Answer Procedures	24
6.5. Removing a Stream with Pseudo-Glare	27
6.5.1. Full Offer/Answer Procedures	27

6.5.2. Partial Offer/Answer Procedures	27
6.6. Changing a Stream with Glare	30
6.6.1. Full Offer/Answer Procedures	30
6.6.2. Partial Offer/Answer Procedures	30
7. Security Considerations	31
8. IANA Considerations	31
9. References	31
9.1. Normative References	31
9.2. Informative References	31
Authors' Addresses	32

1. Introduction

The SDP [RFC4566] offer/answer model defined in [RFC3264] briefly mentions "glare" as a potential issue in the use of offer/answer exchanges, although it relegates the problem to the "higher layer protocol" to resolve. In SIP [RFC3261], resolving state after a glare condition is performed via a timer-based back-off mechanism. For WebRTC, detection of glare comes in the form of an "InvalidStateError" exception. Actual resolution of glare is currently undefined; the present assumption is that the applications that make use of RTCWEB are responsible for handling glare in a sensible fashion.

The penalty for glare isn't simply code complexity; it results in delays in updating sessions state, which can end up visible to users, leading to a less optimal user experience.

Many of the emerging uses for both SIP and RTCWEB involve sessions with a large number of media streams, with streams being added and removed frequently. This kind of session churn increases the incidence of glare significantly.

To reduce the incidence of glare under these circumstances, this document defines a procedure via which partial offer/answer exchanges may take place. These exchanges operate on one or more media sections 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 media sections to contain an "a=mid" [RFC5888] attribute.

This document focuses on the application of this technique for use in RTCWEB and WebRTC. The author anticipates that future work will describe its use in conjunction with SIP and SIP-derived technologies (such as multiparty conferencing and telepresence).

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].

3. Mechanism Overview

The core of this mechanism is the concept of "partial offers" and "partial answers." Syntactically, these entities are SDP fragments, consisting of exactly one o= line; one or more media sections; and any i=, c=, b=, k=, and a= lines associated with the media sections. They are formatted exactly as they would be if they were part of a larger SDP document, with one key exception: unlike SDP, in an SDP fragment, the ordering of media sections relative to each other is not significant. Note that SDP fragments contain only that information that pertains to media. Other than the mandatory o= line, they never contain any session-level information. Within the o= line, only the <sess-version> field is allowed to be changed from its previous value. Any changes to session-level information are expected to use a full offer/answer exchange rather than the partial offer/partial answer mechanism defined by this document.

OPEN ISSUE: Do we need to relax this session-level prohibition? It makes the mechanism clean. However, it does make it difficult to add a new stream while simultaneously associating it with a group. On the other hand, group lines don't have unique identifiers, so just sending a single group line over can be ambiguous. One way around this would be requiring that partial offers and partial answers must contain all group attributes associated with the session, but this still gets potentially messy if we have both sides trying to update group information at the same time. An alternative approach might be to indicate group information only for *new* streams, and require full offers for any other group changes. Of course, without unique group identifiers, we're still stuck with the challenge of unambiguously identifying which group we're adding a new line to. If we constrain group membership so that each group can be uniquely identified by its type and members, then that should be sufficient. Is such a constraint acceptable?

Using this mechanism has two key prerequisites: (1) all offer/answer exchanges in the session prior to sending a partial offer have contained "a=mid" attributes for each media section, 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 an SDP fragment body will be explicitly signaled, e.g., using a different MIME type for SIP, or using a different "type" field for the WebRTC API.

3.1. Adding a Stream

To add a stream glarelessly, a party creates a "partial offer" consisting of an o= line and one or more media sections, including all of the corresponding i=, c=, b=, k=, and a= lines. Each media section contains an "a=mid" attribute, indicating an MID that has not yet been used in the session.

Upon receipt of a partial offer, an implementation processes each media section independently. For each media section, the recipient 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 media section is simply appended to the end of the existing session description, the SDP sess-version is increased, and an answering media section is created. Once all answering media sections have been processed, they are concatenated into a partial answer. This partial answer consists of one or more media sections, each containing 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 media sections within the session description. 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 among each media section's MID attribute: the media sections are appended to the session in lexically increasing order.

3.2. Changing a Stream

Partial offers may also be generated for modification of an existing stream. In this case, the MID in the media section of a 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 media section. Assuming the recipient does not have an outstanding, unanswered partial offer that also modifies that exact same stream, this media section is treated as an independent renegotiation of that stream (only). The SDP

version is increased, and a partial answer is created. This partial answer consists of an media section and its attributes, and has an MID matching the one from the partial offer.

OPEN ISSUE: Since stream *changes* can result in glare, the foregoing text assumes that only one media section will be sent for such a change. Is this okay?

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.

3.3. Removing a Stream

To remove one or more a streams in a way that eliminates the chance of glare, an implementation generates a new partial offer, containing one or more media sections. Each media section contains an MID matching the stream it wants to remove, and indicates a transport port of zero, 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 a port number of zero, then the stream is deactivated. At this point, both partial offers are discarded, the corresponding media section in the session is modified by changing its port to zero, and a partial answer is generated representing this single change.

4. Use With Other Protocols

Note that this document simply defines the extensions to the SDP offer/answer model for dealing with partial offers and partial answers. In the same way that [RFC3264] does not define specific SIP, JSEP, or WebRTC handling, neither does this document. In order for this technique to be useful, protocol-specific mechanisms need to be defined. This additional work is left to appropriate venues, such as the W3C WebRTC WG, the RTCWEB WG, and the SIPCORE WG. If the higher-level protocol allows the use of unordered message delivery, it is that protocol's responsibility to ensure that the result of partial offer/partial answer exchanges is a shared and identical session state between the parties involved.

To assist in understanding the mechanism being proposed, we describe, in a very high-level and non-normative way, how this mechanism might be applied to a couple of specific higher-level signaling systems.

4.1. High-Level Sketch: Use With JSEP/WebRTC

For WebRTC, we envision that such additional specification would add a new constraint to `createOffer`, requesting that a partial offer be generated (if possible). The resulting `RTCSessionDescription` would contain only the media sections 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 media sections referenced in the partial offer, that can be provided to the remote party.

4.2. High-Level Sketch: Use With SIP

For SIP, partial offers and partial answers will likely be provided in SIP UPDATE [RFC3311] or INFO [RFC6086] messages, containing a special "application/sdpfrag" MIME type [I-D.ivov-dispatch-sdpfrag], and a content-disposition that indicates that the contents are a partial offer (rather than, say, a trickle ice candidate). Although INVITE may seem like a natural fit for this kind of behavior, its current definition includes strong glare resolution behaviors that makes it unsuitable for this purpose. Naturally, any such mechanism will be paired with a SIP feature tag that allows for negotiation of support for partial offers and answers.

5. Protocol Operation

The following sections formally defines the procedures for generating and processing partial offers and partial answers.

At any time during an ongoing session, either agent in the session MAY generate a new partial offer that updates the session, subject to the restrictions described in the following sections. However, it MUST NOT generate a new partial offer if it has received any partial or full offer which it has not yet answered or rejected.

An agent also MUST NOT generate a partial offer if it has sent a partial or full offer which has not yet been accepted or rejected.

OPEN ISSUE: It seems like we might be able to have multiple outstanding sent partial offers at once, as long as they don't try to act on the same media section. The reason it's disallowed in the above paragraph is that having several partial offers potentially outstanding in both directions makes it very, very, very complicated to resolve the ordering of media sections if these partial offers in opposite directions overlap temporally.

In the situations described as "glare" below, the higher layer protocol needs to provide a means for resolving such conditions. This will generally be the same mechanism used to resolve the glare conditions described in [RFC3264].

5.1. Common Procedures

For all of the procedures described in the following sections, whenever an o= line is included in a partial offer or partial answer, its <username>, <sess-id>, <nettype>, <addrtype>, and <unicast-address> values MUST be identical to those sent in the most recent full offer or full answer generated by this agent for this session. The <sess-version> value MUST be larger than the value in all previously sent offers, partial offers, answers, and partial offers generated by this agent for this session.

Whenever the procedures in the following sections indicate that a media section is to be included in a partial offer or partial answer, that media section MUST consist of an m= line along with all i=, c=, b=, k=, and a= lines associated with that media section. If a line is absent from a media section in a partial offer or partial answer, it MUST be interpreted as an explicit removal of that value from the media section. Recipients of such messages MUST NOT assume that a previously-established but omitted value is still in effect.

5.2. Generating a Partial Offer

Whenever an agent wishes to change the state of the media in an ongoing session -- whether through addition, modification, or removal of a stream -- it does so through either an offer or a partial offer. In deciding which to use, the implementation first verifies that it has received positive confirmation that the remote implementation supports the partial offer/partial answer mechanism. The means of negotiating such support is left to the higher-level protocol that makes use of the offer/answer model. The implementation then verifies that all media sessions in the current session are associated with unique MID values. Finally, the implementation evaluates whether the changes it needs to make can be performed exclusively using the values present in a media section, without any modifications necessary to session-level values (except for the sess-version value on the session-level o= line).

If all three of the criteria described above are true, then the implementation MAY send a partial offer to make the changes it wants to request. If any of these criteria are not true, then the implementation MUST use a full offer, according to the procedures described in [RFC3264].

Once the agent determines that the change it wishes to make is eligible to use the partial offer mechanism, it forms a new SDP fragment by following these steps:

1. For each desired change to an existing media section, the agent creates a new partial offer consisting of one o= line and a single media section. This media section MUST contain an "a=mid" attribute containing an MID that matches the media section that is being modified. The media section also contains the modifications that the agent wishes to make, as described in section 8.3 of [RFC3264]. This partial offer is sent in isolation, with no other media section changes, additions, or removals in the same partial offer.
2. If the desired change involves one or more media section additions or removals, the agent creates a new partial offer consisting of one o=line and any media section described in the following two steps.
3. For each new media section to be added, the agent creates a new media section to be added to the aforementioned partial offer. This media section MUST contain an "a=mid" attribute, and the MID present in this attribute MUST contain at least 32 characters chosen randomly from full set of 79 characters allowed in a token. The remainder of the media section contains the various values that the agent wishes to have associated with the corresponding media, and is created according to the procedures described in section 5.1 or 5.2 of [RFC3264], as appropriate.
4. For each existing media section to be removed, the agent creates a new media section to be added to the aforementioned partial offer. This media section MUST contain an "a=mid" attribute containing an MID that matches the media section that is being removed, and MUST contain a <port> value of 0 (zero). Except for the required "a=mid" attribute, this media section MAY omit any or all i=, c=, b=, k=, and a= lines, and MAY list only one m= line <fmt> value.

Once the preceding steps have been followed to create one or more partial offers, the agent makes use of the high-level signaling protocol to convey the offers to the remote agent, one at a time.

5.3. Processing a Partial Offer

Upon receipt of a partial offer, an agent first determines whether it has sent any full offers for the corresponding session. If it has, then the partial offer represents a glare condition that is resolved via the higher-level protocol. It then verifies whether it has

received any partial or full offers to which it has not yet sent an answer or a rejection. If so, then it rejects the partial offer as invalid behavior.

The agent then examines the o= line in the received partial offer. If the <sess-version> value is less than the most recently received full (non-partial) offer or answer, then the partial offer is stale and MUST be rejected. The means for rejecting the partial offer are left to the higher-level protocol.

After such validation takes place, the agent iterates through each media section and performs the following steps:

1. If the MID present in the received media section matches a media section already present in the ongoing session and has a non-zero port number, it represents a change to an existing media stream.
 - * If the partial offer contains more than one media section, then the recipient MUST reject the partial offer as invalid behavior.
 - * If the MID matches the MID of a media section in a partial offer that the agent has sent, AND the sent media section contains a port number of zero, then the incoming partial offer is rejected, as any such changes have been "overtaken by events:" the stream will be deactivated momentarily.
 - * The recipient verifies that the MID does not match the MID of any media section in any partial offers that it has sent but has not yet received a partial answer or rejection for, unless the media section in the sent partial offer has a port number of zero. If this verification fails, then the received partial offer represents a glare condition that is resolved via the higher-level protocol.
 - * After the preceding verifications have succeeded, the agent creates media section to include in the partial answer. To reject the media section in the partial offer, the agent generates a media section with a port number set to zero; otherwise, the agent forms the media section by following the procedures described in section 6.1 or 6.2 of [RFC3264], as appropriate.
2. If the MID present in the received media section matches a media section already present in the ongoing session and has a port number of zero, then it represents the removal of an existing media stream. The agent creates a media section to include in the partial answer. With the exception of the "a=mid" attribute,

this media section MAY omit any or all i=, c=, b=, k=, and a= lines, and MAY indicate a single payload type.

3. If the MID present in the received media section does not match any media section already present in the ongoing session, then it represents a new media stream.
 - * If the received media section contains a port number of zero, then the recipient MUST reject the partial offer as invalid behavior: this mechanism does not support the atomic addition and removal of the same stream.
 - * If the above validation succeeds, the agent creates a media section to include in the partial answer. To reject the media section in the partial offer, the agent generates a media section with a port number set to zero; otherwise, the agent forms the media section by following the procedures described in section 6.1 or 6.2 of [RFC3264], as appropriate.

All media sections that are formed in the foregoing steps MUST contain an "a=mid" attribute matching the MID that was present in the corresponding media section from the partial offer.

If the preceding steps have been performed for each media section without resulting in a rejection, then the agent forms a partial answer consisting of a single o= line, and all of the media sections that were generated as part of the preceding steps. Note that this processing will always yield the same number of media sections in a partial answer as were present in the partial offer. Unlike normal SDP processing, however, the order of the media sections in a partial answer is not significant. This partial answer is then sent to the remote agent using the high-level protocol.

If the above processing results in a successful partial answer, then the agent's view of the session is updated as described in Section 5.5.

5.4. Processing a Partial Answer

When a partial answer is received, the offerer matches each media section in the partial answer to its corresponding media section according to its MID. The agent MUST NOT assume that the order of the sections in the received partial answer matches the order of the sections it sent in the partial offer. However, it can expect that each section in the partial offer has a corresponding section in the receive partial answer.

For each media section, the agent then updates its local view of session state as described in Section 5.5, and follows the process described in section 7 of [RFC3264].

5.5. Updating the Shared View of Session State

Whenever a partial offer or partial answer is processed, the agent performs the following steps to ensure that a common view of session state is maintained:

1. The remote session's <sess-version> value is updated according to the value received in the o= line of the sdpfrag.
2. Any changed or removed media sections are modified in-place. Their position in the overall session description remains the same as it was before.
3. Any added media sections are appended to the existing session. The order in which they are appended is determined by lexically sorting them according to their MID values. This is not necessarily the same order in which they appear in the sdpfrag. If the recipient of a partial offer had a sent a partial offer to which it had not yet received a response when the partial offer was received, then it must take additional steps to ensure a common view of the media section ordering: the media sections for the sent partial offer and the received partial answer are treated as a single list, sorted lexically according to their respective MID values, and appended to the session in that order. When that agent receives the corresponding partial answer, the media section ordering remains the same as was established by the partial offer.

Note that this requires the ability to re-index media sections in the case that the remote party rejects the outstanding partial offer that we sent them (I don't mean declining the line by setting the port to zero; I mean that the higher-level protocol actually rejected the offer, like getting an unrecoverable 400- or 500-class error in SIP).
OPEN ISSUE: This is kinda ugly. Is there maybe some better way to handle the issue?

5.6. Receiving a Full Offer with a Partial Offer Pending

For completeness, this document notes that an agent that receives a full offer with a sent partial offer pending is in a glare condition; this is resolved via the higher-level protocol.

6. Examples

The SDP examples given in these examples 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.

For the use-cases that follow, a full Offer/Answer SDP is shown followed by application of the procedures defined in this document to generate Partial Offers and Answers in carrying out the use-case.

As a pre-condition, the SDP below represents the stable state of system after a successful [RFC3264] Offer/Answer negotiation to setup a communication session with one audio (G.711) and one video (VP8) stream.

This SDP serves as base SDP for generating Offers/Partial Offers and shall be terms as Base-SDP going forward.

```
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
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
```

```
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 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxV
a=rtpmap:98 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

6.1. Adding Streams

6.1.1. Full Offer/Answer Procedures

The following SDP shows an offer that adds an audio media section with Opus codec to the Base-SDP:

```
v=0
o=- 20518 1 IN IP4 198.51.100.1          // Version number is incremented
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 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host

m=audio 55800 RTP/SAVPF 109              // New audio media line for opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
```

```
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

The following shows answer for the above Offer accepting the changes:

```
v=0
o=- 20518 1 IN IP4 198.51.100.2      // Version number is incremented
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 0
a=mid:4TOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.2 60401 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxv
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host

m=audio 60604 RTP/SAVPF 109          // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60604 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60605 typ host
```

6.1.2. Partial Offer/Answer Procedures

In order to add an audio media section with Opus codec the Offerer generates the following Partial Offer:

```
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
m=audio 55800 RTP/SAVPF 109          // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
```

```
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

On receiving the above Partial Offer, the Answerer follows the validations defined in the Section 5.3 to generate a Partial Answer. Since the content of the "a=mid" attribute doesn't match any existing values and the Port Number is non zero, thus generated Partial Answer reflects accepting the new audio stream.

Below shows Partial Answer generated by the Answerer in response to the above Partial Offer.

```
o=- 20518 1 IN IP4 198.51.100.2
m=audio 60604 RTP/SAVPF 109 // Answerer accepts the new media stream.
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60604 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60605 typ host
```

On successful Partial Offer/Answer exchange, the Offerer appends the media section offered in the Partial Offer to its Base-SDP. Also <sess-version> is updated as per o= line received. Updated SDP at the Offerer is shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number updated
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 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host

m=audio 55800 RTP/SAVPF 109 // Appended per Partial Offer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

Identical steps are performed by the Answerer on the media section in the Partial Answer as shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2 // Version number updated
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 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.2 60601 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host

m=audio 60604 RTP/SAVPF 109 // Appended per Partial Answer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60604 typ host
```

```
a=candidate:1 2 UDP 2113667326 203.0.113.2 60605 typ host
```

6.2. Removing Streams

6.2.1. Full Offer/Answer Procedures

The following SDP shows an offer that removes the audio media section with PCMU Codec from the Base-SDP:

```
v=0
o=- 20518 1 IN IP4 198.51.100.1          // Version number is incremented
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 0 RTP/SAVPF 0                    // Port is set to zero.
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwThlJYRRNORarcTbGl1QxV
a=rtpmap:98 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

The following shows answer for the above Offer accepting the changes:

```
v=0
o=- 20518 1 IN IP4 198.51.100.2          // Version number is incremented
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 0 RTP/SAVPF 0 // Removal of stream is accepted
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.2.2. Partial Offer/Answer Procedures

In order to remove the audio media section from the Base-SDP the Offerer generates the following Partial Offer:

```
o=- 20518 1 IN IP4 198.51.100.1 // Version number incremented
m=audio 0 RTP/SAVPF 0 // Port is set to zero
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW // mid attribute is included
a=rtpmap:0 PCMU/8000
```

On receiving the above Partial Offer, the Answerer follows the validations defined in the Section 5.3 to generate a Partial Answer that accepts the removal of the corresponding media section.

Below shows the Partial Answer generated by the Answerer in response to the above Partial Offer.

```
o=- 20518 1 IN IP4 198.51.100.2
m=audio 0 RTP/SAVPF 0 // Removal of stream is accepted
a=mid:AOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
```

On successful Partial Offer/Answer exchange, the Offerer updates its Base-SDP to reflect removing of the audio media stream. Also <sess-version> is updated as per o= line received. Updated SDP at the Offerer is shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number updated
```

```
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 0 RTP/SAVPF 0 // Port is updated to zero
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000

m=video 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:98 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

Identical steps are performed by the Answerer on the media section in the Partial Answer as shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2 // Version number updated
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 0 RTP/SAVPF 0 // Port is updated to zero
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.3. Changing a Stream

6.3.1. Full Offer/Answer Procedures

The following SDP shows an offer that marks video stream as sendonly:

```
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number is incremented
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 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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 55600 RTP/SAVPF 120
a=mid:ONy4mOBV2MwTHlJYRRNORarcTbG1lQxv
a=rtpmap:120 VP8/90000
a=sendonly // Video stream is marked as sendonly.
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

The following shows answer for the above Offer accepting the changes.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2 // Version number is incremented
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 0
a=mid:AOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=recvonly // Answerer accepts the change and marks
            // the stream as recvonly.
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.3.2. Partial Offer/Answer Procedures

In order to mark video stream as sendonly, an Partial Offer is generated:

```
o=- 20518 1 IN IP4 198.51.100.1 // Version number is incremented
m=video 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendonly // Video stream is marked as sendonly.
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

Since the content of "a=mid" attribute in the Partial Offer matches, the Answerer generates an Partial Answer with media section corresponding to the video stream accepting the changes, as shown below.

```
o=- 20518 1 IN IP4 198.51.100.2
m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=recvonly // Answerer accepts the change and marks
            // the stream as recvonly on the Answer.
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

On successful Partial Offer/Answer exchange, the Offerer updates the video media section by changing the direction attribute to sendonly. Also <sess-version> is updated as per o= line received,

```
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number updated
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 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendonly // direction updated as per Partial O/A exchange
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

Identical steps are performed by the Answerer on the media section in the Partial Answer as shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2 // Version number updated
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 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=recvonly // direction updated as per Partial O/A exchange
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.4. Both Sides Simultaneously Add Streams

Let Alice and Bob be the peers communicating. In this scenario both the parties attempt to add a new media stream at the same time.

6.4.1. Full Offer/Answer Procedures

This scenario results in the glare situation and should be resolved by the higher-level protocol.

6.4.2. Partial Offer/Answer Procedures

Alice sends a Partial Offer, shown below, to add an audio media section for Opus Codec.

```
o=- 20518 1 IN IP4 198.51.100.1 // Version number is incremented
m=audio 55800 RTP/SAVPF 109 // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

At the same time, Bob sends the following Partial Offer to add an video media section for H.264 Codec.

```
o=- 20518 1 IN IP4 198.51.100.2 // Version number is incremented
m=video 60604 RTP/SAVPF 99 // New video media line for H.264
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT
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 60604 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60605 typ host
```


On receiving the Partial Offer from Bob, Alice verifies from the content of "a=mid" value as an indication of new media being added. It generates the Partial Answer accepting Bob's request to add the new video stream.

```
o=- 20518 1 IN IP4 198.51.100.1
m=video 55900 RTP/SAVPF 99          // Alice accepts Bob's Partial Offer
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT // MID from Partial Offer
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 55900 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 55901 typ host
```

Symmetrically Bob carries out similar actions on Alice's Partial Offer and generates an Partial Answer as shown below.

```
o=- 20518 1 IN IP4 198.51.100.2
m=audio 60606 RTP/SAVPF 109        // Bob accepts Alice's Partial Offer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW // MID from Partial Offer
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60606 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60607 typ host
```

On successful Partial Offer/Answer exchange, Alice appends to the Base-SDP, the two media sections that correspond to audio and video streams negotiated as part of aforementioned Partial Offer/Answer exchanges. Also <sess-version> is incremented to reflect the shared state. The media sections are appended in the lexically increasing order.

```
v=0
o=- 20518 2 IN IP4 198.51.100.1 // Version number updated
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 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

```
m=audio 55800 RTP/SAVPF 109 // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

```
m=video 55900 RTP/SAVPF 99 // Alice accepts Bob's Partial Offer
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT
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 55900 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 55901 typ host
```

Similarly, below SDP shows Bob's Base-SDP updated.

```
v=0
o=- 20518 2 IN IP4 198.51.100.2 // Version number updated
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 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
```

```
a=candidate:1 2 UDP 2113667326 192.0.2.2 60601 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host

m=audio 60606 RTP/SAVPF 109 // Bob accepts Alice's Partial Offer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60606 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60607 typ host

m=video 60604 RTP/SAVPF 99 // New video media line for H.264
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT
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 60604 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60605 typ host
```

6.5. Removing a Stream with Pseudo-Glare

In this example, Alice attempts to change the direction of the video stream to recvonly and Bob attempts to de-activate the same video stream simultaneously.

6.5.1. Full Offer/Answer Procedures

This scenario results in the glare situation and should be resolved by the higher-level protocol.

6.5.2. Partial Offer/Answer Procedures

The term pseudo-glare signifies those scenarios wherein both parties attempt to operate on a stream at the same time, but the final session state can be unambiguously resolved by both sides without any further signaling.

Such an scenario arises when one side tries to change a media section and simultaneously the other party attempts to remove that media section.

Below represents the Partial Offer from Alice to change the direction attribute of the video section to recvonly.

```
o=- 20518 1 IN IP4 198.51.100.1 // Version number is incremented
m=video 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=recvonly // direction changed to recvonly
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

At the same time, Bob sends the following Partial Offer to remove the same media section:

```
o=- 20518 1 IN IP4 198.51.100.2 // Version number is incremented
m=video 0 RTP/SAVPF 120 // Port number is set to 0.
a=mid:0Ny4mOBV2MWTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
```

On validating the Partial Offer from Alice, Bob concludes the media section matches the one in the Partial Offer sent by him. By following the procedures in Section 5.3, Bob rejects Alice Partial Offer since the video media section will be disabled momentarily.

Bob sends the Partial Answer by setting port number to zero as response to Alice's Partial Offer.

```
o=- 20518 1 IN IP4 198.51.100.1 // Version number is incremented
m=video 0 RTP/SAVPF 120 // Alice's Partial Offer rejected
a=mid:0Ny4mOBV2MWTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
```

The processing by Alice is insignificant, because it will eventually be overtaken by Bob's rejection of her Partial Offer and thus the Partial Offer/Answer exchange concludes by removing the video section.

Finally the Base-SDPs are updated by both the parties ending up in the shared state.

```
// Alice's Base-SDP updated
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number updated
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 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
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 120 // Video stream removed
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host


// Bob's Base-SDP
v=0
o=- 20518 1 IN IP4 198.51.100.2 // Version number updated.
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 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.2 60601 typ host

m=video 0 RTP/SAVPF 120 // Port is zero per Bob's Partial Offer
a=mid:0Ny4mOBV2MwTHlJYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
```

```
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.6. Changing a Stream with Glare

In this example both Alice and Bob attempt to update the same media section that conflicts each others actions.

6.6.1. Full Offer/Answer Procedures

This scenario results in the glare situation and should be resolved by the higher-level protocol.

6.6.2. Partial Offer/Answer Procedures

To explain this scenario, say Alice attempts to update the video section's direction to be sendonly and Bob also attempts to perform the same action.

This results in a conflict situation since both the parties can't have the same media section with sendonly direction, since it violates rules defined in [RFC3264]

Partial Offer's generated by Alice and Bob for the above scenario is shown below.

```
// Alice's Partial Offer
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
m=video 55600 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendonly                          // direction changed to sendonly
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

```
// Bob's Partial Offer
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MwTH1JYRRNORarcTbG1lQxV
a=rtpmap:120 VP8/90000
a=sendonly                          // direction changed to sendonly
```

```
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

This results in a glare situation under Partial Offer/Answer exchange due to the conflicting nature of the actions. To resolve this situation assistance from the higher level application protocol is required.

7. Security Considerations

TBD

8. IANA Considerations

TBD -- I don't think we actually need any for this mechanism.

9. References

9.1. Normative References

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

9.2. Informative References

- [I-D.ivov-dispatch-sdpfrag] Ivov, E. and A. Roach, "Internet Media Type application/sdpfrag", draft-ivov-dispatch-sdpfrag-03 (work in progress), October 2013.

[RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[RFC6086] Holmberg, C., Burger, E., and H. Kaplan, "Session Initiation Protocol (SIP) INFO Method and Package Framework", RFC 6086, January 2011.

Authors' Addresses

Adam Roach
Mozilla
Dallas, TX
US

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

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2014

M. Thomson
Microsoft
October 19, 2013

Using Interactive Connectivity Establishment (ICE) in Web Real-Time
Communications (WebRTC)
draft-thomson-mmusic-ice-webrtc-01

Abstract

Interactive Connectivity Establishment (ICE) has been selected as the basis for establishing peer-to-peer UDP flows between Web Real-Time Communication (WebRTC) clients. Using an unmodified ICE implementation in this context enables the use of the web platform as a denial of service platform. The risks and complications arising from this choice are discussed. A modified algorithm for sending ICE connectivity checks from the web platform is described.

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 April 22, 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
1.1. Conventions and Terminology	4
2. ICE in a Web Browser	4
2.1. Factors Influencing DoS Capacity	4
2.1.1. Pacing of Connectivity Checks	5
2.1.2. Retransmission of Connectivity Checks	5
2.1.3. Connectivity Check Size	6
2.2. Denial of Service Magnitude	6
3. Modified ICE Algorithm	7
3.1. Trickle and Peer Reflexive Candidates	9
3.2. Multiple ICE Agents	10
3.2.1. Introducing Artificial Contention	11
3.2.2. Origin-First Round-Robin	11
3.2.3. Inter-Agent Candidate Pair Freezing	11
3.2.4. Delayed ICE Agent Start	12
4. Further Reducing the Impact of Attacks	12
4.1. Bandwidth Rate Limiting	12
4.2. Malicious Application Penalties	13
4.3. Limited Concurrent Access to ICE	13
5. Negotiating Algorithm Use	13
6. Security Considerations	14
7. Acknowledgements	14
8. References	14
8.1. Normative References	14
8.2. Informative References	14
Appendix A. Defining Legitimate Uses of ICE	15
A.1. Candidate Pair Count	15
A.2. Connectivity Check Size	16
A.3. Rate Calculations	16
A.4. Comparison: G.711 Audio	16
A.5. Recommended Rate Limits	17
Author's Address	17

1. Introduction

ICE [RFC5245] describes a process whereby peers establish a bi-directional UDP flow. This process has been adopted for use in Web Real-Time Communications (WebRTC) for establishing flows to and from web browsers ([I-D.ietf-rtcweb-overview]).

Properties of ICE are also critical to the security of WebRTC (see Section 4.2.1 of [I-D.ietf-rtcweb-security]).

The design of RFC 5245 does not fully consider the threat models enabled by the web environment. In particular, the following assumptions are not valid in a web context:

- o A one-time consent to communicate is sufficient, and revocation of consent is not necessary.
- o Signaling and control originates from actors that always operate in good faith.
- o Only one ICE processing context operates at the one time.

Implementations of ICE that are technically compliant with the algorithm described in RFC 5245 potentially expose controls to web applications that can be exploited.

In the web context, an attacker is able to provide code (usually JavaScript) that is executed by those hosts in a sandbox. The protections of the sandbox are critical, both for protecting the host running the sandbox, and for protecting the Internet as a whole from bad actors.

The exposure of ICE features in the web browser could allow attackers to generate denial of service (DoS) traffic far in excess of the bandwidth needed to deploy the JavaScript. A small (1KB) file can potentially generate many megabytes of connectivity checks in a short period, representing an amplification factor far greater than other similar amplification attacks (for instance, DNS reflection attacks).

Mounting this sort of DoS attack does not rely on anything other than inducing a host to download and execute JavaScript. This is generally very easy to accomplish, making it very easy to conscript large number of traffic sources.

The issue regarding the one-time consent to communicate has already been identified as a serious problem for WebRTC. [I-D.muthu-behave-consent-freshness] describes a limit on the time that consent remains valid, requiring that communications consent be continuously refreshed.

This document first describes the characteristics of ICE as they relate to the web and the way that these characteristics can be exploited. In order to address the issues arising from allowing web application to initiate and control ICE processing, a modified algorithm is described, plus additional measures that can be employed to reduce the amount of traffic an attacker can produce.

1.1. Conventions and Terminology

In cases where normative language needs to be emphasized, this document falls back on established shorthands for expressing interoperability requirements on implementations: the capitalized words "MUST", "MUST NOT", "SHOULD" and "MAY". The meaning of these is described in [RFC2119].

2. ICE in a Web Browser

A web browser provides an API that applications can use to instantiate and control an ICE agent. The web application is responsible for providing the ICE agent with signaling that it might need to operate successfully, as well as configuration information regarding TURN [RFC5766] or STUN [RFC5389] servers.

In the web context, a browser treats the web application as being potentially hostile, providing access to features in a controlled fashion. Therefore, some of the information that an ICE agent might depend on in other contexts has to be regarded as potentially suspect when provided by a web application.

2.1. Factors Influencing DoS Capacity

There are several parameters that affect the characteristics of DoS attacks that can be mounted using ICE. These include:

- o The number of candidate pairs that are created. An attacker can add extra remote candidates to inflate this number to the maximum supported. RFC 5245 recommends a default maximum of 100 candidate pairs. Reducing this limit directly reduces DoS potential, though it could affect success in some legitimate scenarios (see the calculations in Appendix A).
- o The time between consecutive connectivity checks. Pacing of checks is discussed at length in Section 2.1.1.
- o The total number and timing of retransmissions for each candidate pair. Section 2.1.2 discusses the implications of retransmissions.
- o The size of connectivity check packets. Size considerations are described in Section 2.1.3.
- o The number of ICE agents that can be operated concurrently. RFC 5245 does not consider scenarios like WebRTC where it is not only possible for there to be multiple agents. The web security model allows for cases where multiple agents can be created

concurrently, often with a further restriction that a browser not leak information between agents.

2.1.1. Pacing of Connectivity Checks

ICE [RFC5245] describes a scheme for pacing connectivity checks. There are two primary reasons that are cited:

- o Pacing the initial connectivity checks for a given candidate pair allows middleboxes sufficient time to establish bindings. Empirical evidence suggests that failing to allow at least 20 milliseconds between initial connectivity checks risks the bindings being dropped at some middleboxes.
- o Pacing limits the potential for connectivity checks to generate network congestion. Section 16.1 of [RFC5245] describes a formula for calculating the time between connectivity checks (T_a) that is based on the expected bandwidth of the real-time session that is being established.

In the web context, information about the expected bandwidth used by the session comes from the web application. Since the web application has to be regarded as potentially malicious, information about expected media bandwidth cannot be used to determine the pacing of connectivity checks. A fixed minimum interval between connectivity checks becomes the primary mechanism for limiting the ability of web applications to generate packets that are potentially congestion inducing.

Increasing the pacing interval directly reduces the amount of congestion that connectivity checks can generate, though this only reduces the peak bitrate that can be induced - the same amount of traffic is generated over a longer period. The cost of this is extended session setup times, where recent efforts have been focused on reducing this time.

2.1.2. Retransmission of Connectivity Checks

The initial retransmission timer (RTO) can also be increased with similar effect to increasing the pacing timer. Furthermore, there is a strong desire to reduce the recommended value of the RTO in ICE from 500 milliseconds to values more reflective of common round trip times in well-connected locations, which might be as low as 50 milliseconds.

More relevant is the total number of connectivity check retransmissions that an implementation attempts for each candidate pair. Each additional retransmission directly increases the duration

and magnitude of a DoS attack. Following the exponential backoff recommended by RFC 5245 does extend the time between retransmissions, which could reduce the rate of connectivity checks after several retransmissions, but this depends on the initial retransmission time out (RTO).

Reducing the number of retransmissions has the effect of reducing the probability of the check succeeding. The selection of a total retransmission count is a trade-off of success rates against the potential for abuse.

2.1.3. Connectivity Check Size

As currently specified, an attacker is only able to influence the size of the USERNAME attribute. [RFC5389] restricts USERNAME to a maximum size of 512 octets; the Session Description Protocol (SDP) signaling described in [RFC5245] limits the size of the username fragment an attacker can set to 256 bytes.

A browser could reduce its username fragment to as little as 4 bytes, limiting the overall size of the attribute to 261 bytes. A small username fragment does limit the collision resilience of the field, which is a property that is important for detecting other forms of attack (see Section 5.7.3 of [I-D.ietf-rtcweb-security-arch]).

There is also the potential for new modifications to ICE that increase the packet size. For instance [I-D.martinsen-mmusic-malice] provides an attacker with direct control over the bytes that are included in connectivity checks.

2.2. Denial of Service Magnitude

A malicious application is able to influence connectivity checking by altering the set of remote candidates and by changing the remote username fragment. The default maximum sizes for remote username fragment (256 bytes) and number of candidate pairs (100) described in RFC 5245 can be exploited by an attacker to increase the number and size of packets. Assuming an inter-check timer of the minimum of 20 milliseconds, plus a minimal 28 bytes of IPv4 and UDP overhead, this results in an attacker being able to induce approximately 144kbps for every ICE agent it is able to instantiate.

This rate is significantly higher than the minimal rate of 20kbps that a typical compressed voice stream generates. By comparison, a G.711 audio stream, which cannot be rate limited in response to network congestion, but is generally regarded as safe to send to a willing target, generates about 74kbps.

ICE does not allow for any congestion feedback (other than ECN [RFC3168]), so this rate could conceivably be sustained for some time, though after several seconds the time between retries increases, reducing the check rate unless the application is able to instantiate another ICE agent.

Some existing ICE implementations could generate about 3 or more times the basic rate of connectivity checks over a short period. These implementations do not pace retransmission of connectivity checks, resulting in significantly higher connectivity check rates during early rounds of retransmission.

These implementations are ignoring the advice on calculating a minimum RTO from Section 16.1 of [RFC5245]. However, the shorter RTO allows ICE to complete much faster, which is a significant advantage.

Implementations that do not limit the number of ICE agents that can be instantiated, and subsequently fail to enforce rate limits globally create a further multiplicative factor on the basic rate.

3. Modified ICE Algorithm

This section describes an algorithm that ensures proper global pacing of connectivity checks. This limits the ability of any single attacker to generate a high rate of connectivity checks. This only limits the peak data rate that results from connectivity checks, reducing the intensity of DoS attacks.

Measures that reduce the overall duration of attacks are described in Section 4.

The modified algorithm for ICE does not alter the way that candidate pairs are selected, prioritized, frozen or signaled. It only affects the generation of connectivity checks. This algorithm affects candidate pairs in either of the "Waiting" or "In-Progress" states only (see Section 5.7.4 of [RFC5245]).

The ICE agent maintains two queues for candidate pairs.

waiting queue: The first is a prioritized list of candidate pairs in the "Waiting" state. The waiting queue is simply a prioritized list of all the candidate pairs in the check list (see Section 5.7 of [RFC5245]) that are in the "Waiting" state. As candidate pairs enter the "Waiting" state, they are added to the waiting queue. As each candidate pair is added, it is prioritized relative to all the other candidate pairs in the waiting queue.

check queue: The second is for outstanding connectivity checks. Each entry in this list represents a connectivity check for a given candidate pair. Each entry also includes a counter representing the number of connectivity checks that have been sent on this candidate pair.

The ICE agent maintains two types of timer: a pacing timer and a retransmission timer. There is only one pacing timer, though there can be multiple retransmission timers running concurrently.

The first candidate pair that arrives in the waiting queue starts the pacing timer. The pacing timer runs as long as there are items in any queue, ending if the timer expires when there are no entries in either queue. The pacing timer resumes if an entry is added to either queue and the timer is not already running.

Each time the pacing timer expires, the ICE agent performs the following steps:

1. If there are items on the waiting queue, but no items on the check queue, the first candidate pair is taken from the waiting queue.
 - a. The candidate pair transitions from "Waiting" to "In-Progress".
 - b. A check counter is associated with the candidate pair, initialized with a zero value.
 - c. The candidate pair is added to the check queue. This could result in a connectivity check being sent immediately if the check queue is currently empty.
2. If there are items in the check queue, the ICE agent removes the first item and performs a connectivity check on the identified candidate pair.
 - a. The check counter associated with the candidate pair is incremented by one.
 - b. Based on the value of the check counter, a retransmission timer is scheduled for the candidate pair. The retransmission timer is not scheduled if the check counter exceeds the maximum number of checks configured for the ICE agent.

- c. If the retransmission timer expires without the connectivity check succeeding, the candidate pair is returned to the end of the check queue along with the higher check counter.
 - d. The retransmission timer is cancelled if the connectivity check succeeds. The process for handling successful checks in Section 7.1.3.2 of [RFC5245] is followed.
3. If no connectivity checks were sent, the pacing timer is stopped.

An important characteristic of this algorithm is that it - as much as possible - prefers retransmission of connectivity checks over the initiation of new connectivity checks. This ensures that once an initial connectivity check has established any necessary middlebox bindings, subsequent retries are not delayed excessively, which could cause the binding to time out. However, the global pacing can cause the time between retransmission of connectivity checks to be extended as the check queue occasionally fills.

Favoring retransmission over initial checks directly contradicts the guidance on RTO selection in Section 16.1 of [RFC5245]. This is necessary due to the delays induced by potential interactions between multiple ICE agents, which might otherwise cause retries to be significantly delayed. Improvements to candidate prioritization are expected to reduce the impact of this change.

3.1. Trickle and Peer Reflexive Candidates

Trickle ICE candidates [I-D.ivov-mmusic-trickle-ice] generate candidate pairs after connectivity checking has commenced. In order to avoid trickle candidates negatively affecting the chances of a connectivity check succeeding, connectivity checks on newly appearing candidate pairs must be prioritized below any existing connectivity check.

Trickle candidates are in many respects identical to peer reflexive candidates. Both arrive after the algorithm has commenced.

In either case, as new candidates arrive (or are discovered), they are paired as normal (Section 5.7.1 of [RFC5245]), and - if appropriate - entered into the "Waiting" state. This causes the candidate pair to enter the waiting queue. Candidate pairs in the waiting queue are not ordered based on arrival time, they are ordered based on priority alone.

Trickling regular candidates does introduce the potential for a mismatch in the ordering of candidate pairs between peers, since trickle candidates will appear in the sending side well before the

receiving side can act upon them, resulting in the sending peer potentially commencing checks much earlier than the receiving peer. This is particularly important given the possibility that retransmissions of connectivity checks can block the progress of a candidate pair from the "Waiting" state into the "In-Progress" state, resulting in potentially large differences in the commencement time for any given candidate pair.

A trickle ICE implementation MAY choose not to immediately enqueue local candidates as they are discovered to allow some time for trickle signaling to propagate in order to increase the probability that checks remain synchronized.

3.2. Multiple ICE Agents

In a system that has potentially more than one ICE agent, it's important that connectivity checks from any given ICE agent cannot be blocked or starved by other ICE agents. It is also important that an attacker is unable to circumvent any limits by instantiating multiple ICE agents.

To that end, a single pacing timer is maintained globally whenever multiple ICE agents are operated. Each time the pacing timer fires, the global context selects ICE agents in a round-robin fashion. In addition to ensuring a global rate limit, this selection method ensures that no single ICE agent is completely starved.

In a shared context, ICE agents do not stop or start the pacing timer unless they are the first or last ICE agent to be active. The first ICE agent to commence checking starts the global timer, the last ICE agent to cancel the timer causes the global timer to be cancelled. At all other instances, "starting" the pacing timer for an ICE agent simply adds the ICE agent to the set of agents that can be selected; "stopping" the pacing timer removes the ICE agent from the set of ICE agents that are in consideration.

A global pacing timer causes each individual ICE agent to execute checks more slowly than a lone ICE agent would. Where there are many candidate pairs to test, this could have a negative impact on the synchronization of checks between peers. Poor check synchronization can have a negative impact on success rates. Peers with asymmetric contention can have lower priority candidate pairs started on the less contended peer long before the contended peer is able to commence checking, which can result in those checks failing.

Several measures are suggested for mitigating the impact of contention: artificial contention, origin-first distribution, inter-

agent candidate pair freezing, and delayed start. However, it is important to note that similar artificial constraints have classically been quickly circumvented on the web if they have overly negative performance consequences.

3.2.1. Introducing Artificial Contention

In cases where there is zero contention, artificial contention can be introduced to ensure a certain minimum effective pacing timer. In effect, this would increase the basic pacing timer from 20ms by a minimum multiple for any single ICE agent. Artificially contention would result in no checks being sent at all at different phases, spacing genuine connectivity checks.

For instance, contention could be increased to a minimum of 3 ICE agents. Assuming a 20ms basic interval, the first ICE agent would be able to send connectivity checks every 60ms, as though it were contending with two other ICE agents. Adding another ICE agent would have no effect on this rate. It would only be if a fourth ICE agent were added that all ICE agents would be reduced to sending checks at 80ms intervals.

This has the advantage of ensuring that a lightly contended client has the same rate of checking as a client with only a small number of ICE agents so that checks are more likely to be synchronized.

3.2.2. Origin-First Round-Robin

In a system such as a browser, there are potentially competing interests sharing the same limited resources. In this type of context, each competing user - in the browser, this is an origin [RFC6454] - can first be selected using a round-robin or similar allocation scheme.

Thus, as a first step, selection is performed from the set origins that have an active ICE agent. Once an origin is selected, agents are selected from within that origin. This ensures that no single origin can receive more than a proportional share of the access to connectivity checking.

This is particularly important if multiple users (or origins) are each able to create multiple ICE agents. Selecting based on users first prevents a single origin from monopolizing access to connectivity checks.

3.2.3. Inter-Agent Candidate Pair Freezing

In some cases, it might be necessary to instantiate multiple ICE agents from the same application, between the same two peers. An ICE agent MAY place candidate pairs in the "Frozen" state based on candidate pairs with the same foundation being "Waiting" or "In-Progress" on another ICE agent. This reduces the overall demand for connectivity checks without any significant negative effect on the chances that ICE succeeds.

In the browser context, information about the success of connectivity checks cannot leak between different domains. This could allow information about activities on another tab to be leaked, violating the origin security model of the browser. Thus, any inter-agent freezing logic MUST be constrained to ICE agents that operate in the same origin.

3.2.4. Delayed ICE Agent Start

In cases where there is high contention for access to connectivity checking, it might be preferable to delay the start of connectivity checks for an ICE agent rather than have the effective pacing timer increased.

4. Further Reducing the Impact of Attacks

A global pacing timer allows a web application to determine whether another domain is currently establishing an ICE transport, simply by observing the pacing of connectivity checks that it requests. Section 3.2.1 describes a method that allows a limited number of ICE agents to operate without being detectable.

The algorithm and the measures it describes are based on an assumption that ICE agents are created legitimately. Even with these measures, it's possible to generate a steady amount of bandwidth toward arbitrary hosts. The remainder of this section is dedicated to additional measures that might be employed to reduce the ability of malicious users to generate unwanted connectivity checks over time.

4.1. Bandwidth Rate Limiting

A measure of the bandwidth generated by connectivity checks can be maintained, on both global and a per-origin basis. As this number increases, the browser can reduce the rate of connectivity checks. This reduction might either be gained by increasing the duration of the pacing timer or skipping occasional connectivity checks.

Appendix A includes some simple calculations and recommendations on what might be appropriate limits to set on the bandwidth used by connectivity checks.

4.2. Malicious Application Penalties

An attacker that only wishes to generate traffic is unlikely to provide valid candidates for two reasons:

- o a successful connectivity check is likely to cause the ICE agent to terminate further checking
- o serving connectivity checks requires the dedication of greater resources by the attacker

A long sequence of unsuccessful connectivity checks is therefore a likely indicator for an attack. An ICE agent could choose to reduce the rate at which connectivity checks are generated for an application that has a large number of failed checks.

Any measure that penalizes for unsuccessful checks will have to allow for some failures. Even legitimate uses of ICE can result in significant numbers of failed connectivity checks. For instance, an implementation that exclusively prioritizes IPv6 over IPv4 on a network with broken IPv6 will legitimately see a large number of failures. Similarly, if a remote peer is behind a NAT, prior to the commencement of checking by that peer all connectivity checks are likely to be discarded by the NAT.

4.3. Limited Concurrent Access to ICE

Setting an absolute maximum on the number of ICE agents that can be instantiated could overly constrain legitimate applications that depend on having multiple active sessions. However, limiting concurrent access to active ICE agents by delaying the start of connectivity checking, as described in Section 3.2.4 might allow an implementation to reduce the ability of a single origin to generate unwanted connectivity checks.

5. Negotiating Algorithm Use

The algorithm defined in Section 3 could cause some ICE agents to perform checks in a very different order to the order of an unmodified ICE agent. Failing to coordinate when checks occur reduces the probability that ICE is successful.

TODO: Determine whether an ice-options token that enables negotiation of this algorithm is appropriate, or whether something more

definitive is required, since an answerer could negotiate an ice-options token away. Note that WebRTC implementations probably won't be able to accept a session that does not use this algorithm.

6. Security Considerations

This entire document is about security.

7. Acknowledgements

The bulk of the algorithm described in this document came out of a discussion with Emil Ivov and Pal-Erik Martinsen. Eric Rescorla and Bernard Aboba provided some feedback regarding the DoS considerations and possible mitigations.

8. References

8.1. Normative References

- [I-D.ivorv-mmusic-trickle-ice]
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", draft-ivorv-mmusic-trickle-ice-01 (work in progress), March 2013.
- [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.

8.2. Informative References

- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications", draft-ietf-rtcweb-overview-08 (work in progress), September 2013.
- [I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-07 (work in progress), July 2013.
- [I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-05 (work in progress), July 2013.

- [I-D.martinsen-mmusic-malice]
Penno, R., Martinsen, P., Wing, D., and A. Zamfir, "Meta-data Attribute signaling with ICE", draft-martinsen-mmusic-malice-00 (work in progress), July 2013.
- [I-D.muthu-behave-consent-freshness]
Perumal, M., Wing, D., R, R., and T. Reddy, "STUN Usage for Consent Freshness", draft-muthu-behave-consent-freshness-04 (work in progress), July 2013.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [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.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

Appendix A. Defining Legitimate Uses of ICE

Limiting the bandwidth generated by connectivity checks depends on knowing how much ICE could use under normal circumstances. This ensures any absolute limit doesn't adversely affect a legitimate use of ICE.

Any calculation should allow for slightly abnormal configurations that might generate higher than average data rates. Otherwise, an average might adversely affect legitimate users. The intent is to avoid having legitimate uses concerned with the limit.

A.1. Candidate Pair Count

Our sample legitimate user has 2 local network interfaces. This can result in as many as 14 candidates, 8 of them IPv4 plus 6 IPv6. Each interface has 1 IPv4 address, an IPv6 address, plus a link-local IPv6 address. Assuming a different public IPv4 NAT address for each interface and IP version (using either NAT4-4 or NAT6-4 as appropriate) other than the link local addresses, this adds another 4 addresses. In addition to this, two TURN servers might be contacted by either IPv4 or IPv6, providing 4 more addresses.

Two peers with this configuration will generate 100 candidate pairs, since only IPv4 candidates are paired with IPv4 candidates.

Assuming that all candidates are checked once before ICE completes on a second round of checks, there are in excess of 100 connectivity checks sent. Even at the fastest permitted pacing, this means that ICE completes in at least 2 seconds, plus the round trip time.

A.2. Connectivity Check Size

The STUN message used for a connectivity check can vary, but making some reasonable assumptions, it is likely to be 149 or 169 bytes on the wire (plus network layer encapsulation). This makes the following assumptions:

IP Header: 20 bytes (IPv4) or 40 bytes (IPv6) with no extensions

UDP Header: 8 bytes

STUN Header: 20 bytes

USE-CANDIDATE Attribute: 4 bytes

CONTROLLED or CONTROLLING Attribute: 4 bytes

PRIORITY Attribute: 4 bytes

MESSAGE-INTEGRITY Attribute: 24 bytes

FINGERPRINT Attribute: 8 bytes

USER Attribute: 49 bytes carries two 20 character username fragments

A.3. Rate Calculations

Assuming a 150 byte connectivity check and a global pacing timer of 20ms, this produces 60kbps at peak (68kbps for IPv6).

For 100 candidate pairs, with at most 5 connectivity checks on each pair, this peak could be sustained for 10 seconds by a single ICE agent.

The question is: is this a tolerable rate?

A.4. Comparison: G.711 Audio

G.711 audio is commonly used without any congestion feedback mechanisms in place - primarily because it is unflexible and unable

to scale its network usage in response to congestion signals. The theory is that it might be acceptable to generate a similar amount of traffic without congestion controls.

It should be immediately obvious that this theory has a major flaw. Even though the impact on the network might be similar, G.711 is not sent to an unwilling recipient, whereas no such guarantee can be made for connectivity checks.

Assuming 80bit integrity on SRTP, no header extensions and no CSRCs, G.711 produces 84kbps. That would suggest that a single ICE agent with 20ms pacing might be tolerable, at least over short intervals.

A.5. Recommended Rate Limits

Enforcing a limit of 96kbps would allow for a substantial increase in the size of STUN connectivity check messages without affecting legitimate uses.

Over a longer interval, this high rate is likely to be unnecessary. Even with 100 candidate pairs, ICE should complete in between 2 and 5 seconds, especially if candidate pairs are frozen across multiple ICE agents. Providing a lower limit over a 10 to 20 second interval should further limit the damage. Enforcing a longer term limit of 48 kilobytes (every 20 seconds or so) would allow for 6 seconds of continuous checking with the size described above, or 4 seconds of checking at the short term rate limit.

Author's Address

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

Email: martin.thomson@skype.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2014

M. Westerlund
B. Burman
Ericsson
S. Nandakumar
Cisco
October 22, 2013

Using Simulcast in RTP Sessions
draft-westerlund-avtcore-rtp-simulcast-03

Abstract

In some application scenarios it may be desirable to send multiple differently encoded versions of the same Media Source in independent Source Packet Streams. This is called Simulcast. This document discusses the best way of accomplishing Simulcast in RTP and how to signal it in SDP. A solution is defined by making three extensions to SDP, and using RTP/RTCP identification methods to relate RTP Source Packet Streams. The first SDP extension consists of two new session level SDP attributes that express capability to send or receive Simulcast Source Packet Streams, respectively. The second SDP extension introduces an SDP media level attribute that groups and identifies a selected set of media level parameters for a specific direction, called a media configuration. The third SDP extension describes how to group such media configurations on SDP session or media level for Simulcast purposes.

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 April 25, 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. Definitions	3
2.1. Terminology	3
2.2. Requirements Language	4
3. Use Cases	4
3.1. Reaching a Diverse Set of Receivers	5
3.2. Application Specific Media Source Handling	6
3.3. Receiver Adaptation in Multicast/Broadcast	7
3.4. Receiver Media Source Preferences	7
4. Requirements	8
5. Proposed Solution Overview	9
6. Proposed Signaling	10
6.1. Simulcast Capability	11
6.1.1. Declarative Use	12
6.1.2. Offer/Answer Use	12
6.2. Media Configuration	13
6.2.1. Simulcast Limitations	16
6.2.2. Declarative Use	17
6.2.3. Offer/Answer Use	17
6.3. Grouping Simulcast Configurations	18
6.3.1. Declarative Use	19
6.3.2. Offer/Answer Use	19
6.4. Relating Simulcast Versions	20
6.5. Two-Phase Negotiation	20
6.6. Signaling Examples	21
6.6.1. Unified Plan Client	21
6.6.2. Multi-Transport Client	24
6.6.3. Multi-Source Client	26
7. Network Aspects	28
8. IANA Considerations	29
9. Security Considerations	29
10. Contributors	29
11. Acknowledgements	30

12. References	30
12.1. Normative References	30
12.2. Informative References	31
Appendix A. Discussion on Receiver Diversity	32
Authors' Addresses	34

1. Introduction

Most of today's multiparty video conference solutions make use of centralized servers to reduce the bandwidth and CPU consumption in the endpoints. Those servers receive Source Packet Streams from each participant and send some suitable set of possibly modified streams to the rest of the participants, which usually have heterogeneous capabilities (screen size, CPU, bandwidth, codec, etc). One of the biggest issues is how to perform stream adaptation to different participants' constraints with the minimum possible impact on video quality and server performance.

Simulcast is the act of simultaneously sending multiple different versions of the same media content, e.g. the same video source encoded with different video encoder types or image resolutions. This can be done in several ways and for different purposes. This document focuses on the case where it is desirable to provide a Media Source as multiple Source Packet Streams over RTP [RFC3550] towards an intermediary so that the intermediary can provide the wanted functionality by selecting which Source Packet Stream to forward to other participants in the session, and more specifically how the identification and grouping of the involved Source Packet Streams are done. From an RTP perspective, Simulcast is a specific application of the aspects discussed in RTP Multiplexing Guidelines [I-D.ietf-avtcore-multiplex-guidelines].

The purpose of this document is to describe a few scenarios where it is motivated to use Simulcast, and propose a suitable solution for signaling and performing RTP Simulcast.

2. Definitions

2.1. Terminology

This document makes use of the terminology defined in RTP Taxonomy [I-D.lennox-raiarea-rtp-grouping-taxonomy]. In addition, the following terms are used:

Media Configuration: A specific set of parameter values applied on the encoding and packetization process that creates a specific Source Packet Stream. In SDP, the applicable parameter values are described by the joint set of "rtpmap" parameters, "fmtp"

parameters, and the "config-id" (Section 6.2) parameters, including extensions.

2.2. 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].

3. Use Cases

Many use cases of Simulcast as described in this document relate to a multi-party Communication Session where one or more central nodes are used to adapt the view of the Communication Session towards individual Participants, and facilitate the Media Transport between Participants. Thus, these cases targets the RTP Mixer topology defined in [RFC5117] (Section 3.4: Topo-Mixer), further elaborated and extended with other topologies in [I-D.ietf-avtcore-rtp-topologies-update] (Section 3.6 to 3.9).

There are two principle approaches for an RTP Mixer to provide this adapted view of the Communication Session to each receiving Participant:

- o Transcoding (decoding and re-encoding) received Source Packet Streams with characteristics adapted to each receiving Participant. This often include mixing or composition of Media Sources from multiple Participants into a mixed Media Source originated by the RTP Mixer. The main advantage of this approach is that it achieves close to optimal adaptation to individual receiving Participants. The main disadvantages are that it can be very computationally expensive to the RTP Mixer and typically also degrades media Quality of Experience (QoE) such as end-to-end delay for the receiving Participants.
- o Switching a subset of all received Source Packet Streams or sub-streams to each receiving Participant, where the used subset is typically specific to each receiving Participant. The main advantages of this approach are that it is computationally cheap to the RTP Mixer and it has very limited impact on media QoE. The main disadvantage is that it can be difficult to combine a subset of received Source Packet Streams into a perfect fit to the resource situation of a receiving Participant.

The use of Simulcast is relates to the latter approach, where it is more important to reduce the load on the RTP Mixer and/or minimize QoE impact than to achieve an optimal adaptation of resource usage.

A multicast/broadcast case where the receivers themselves select the most appropriate simulcast version and tune in to the right transport to receive that version is also considered (Section 3.3). This enables large receiver populations with heterogeneity where it comes to capabilities and the use network paths bandwidth.

In this section, an "RTP switch" is used as a common short term for the terms "switching RTP mixer", "source projecting middlebox", and "video switching MCU" as discussed in [I-D.ietf-avtcore-rtp-topologies-update].

3.1. Reaching a Diverse Set of Receivers

The Media Sources provided by a sending Participant potentially need to reach several receiving Participants that differ in terms of available resources. A discussion on that topic is included in Appendix A. The receiver resources that typically differ include, but are not limited to:

Codec: This includes codec type (such as SDP MIME type) and can include codec configuration options (e.g. SDP fmp parameters). A couple of codec resources that differ only in codec configuration will be "different" if they are somehow not "compatible", like if they differ in video codec profile, or the transport packetization configuration.

Sampling: This relates to how the Media Source is sampled, in spatial as well as in temporal domain. 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. This may be used to suit different rendering capabilities or needs at the receiving endpoints, as well as a method to achieve different transport capabilities, bitrates and eventually QoE by controlling the amount of source data.

Bitrate: This relates to the amount of bits spent per second to transmit the Media Source as an Source Packet Stream, which typically also affects the Quality of Experience (QoE) for the receiving user.

Letting the sending Participant create a Simulcast of a few differently configured Source Packet Streams per Media Source can be a good trade-off when using an RTP switch as middlebox, instead of sending a single Source Packet Stream and using an RTP Mixer to create individual transcodings to each receiving Participant.

This requires that the receiving Participants can be categorized in terms of available resources and that the sending Participant can choose a matching configuration for a single Source Packet Stream per category and Media Source.

For example, assume for simplicity a set of receiving Participants that differ only in that some have support to receive Codec A, and the others have support to receive Codec B. Further assume that the sending participant can send both Codec A and B. It can then reach all receivers by creating two Simulcasted Source Packet Streams from each Media Source; one for Codec A and one for Codec B.

In another simple example, a set of receiving Participants differ only in screen resolution; some are able to display video with at most 360p resolution and some support 720p resolution. A sending Participant can then reach all receivers by creating a Simulcast of Source Packet Streams with 360p and 720p resolution for each sent video Media Source.

In more elaborate cases, the receiving Participants differ both in available Sampling and Bitrate, and maybe also Codec, and it is up to the RTP switch to find a good trade-off in which Simulcasted stream to choose for each intended receiver. It is also the responsibility of the RTP switch to negotiate a good fit of Simulcast streams with the sending Participant.

The maximum number of Simulcasted Source Packet Streams that can be sent is mainly limited by the amount of processing and uplink network resources available to the sending Participant.

3.2. Application Specific Media Source Handling

The application logic that controls the Communication Session may include special handling of some Media Sources. It is for example commonly the case that the media from a sending Participant is not sent back to itself.

It is also common that a currently active speaker Participant is shown in larger size or higher quality than other Participants (the Sampling or Bitrate aspects of Section 3.1). Not sending the active speaker media back to itself means there is some other Participant's media instead that receive special handling towards the active speaker; typically the previous active speaker. This way, the previously active speaker is needed both in larger size (to current active speaker) and in small size (to the rest of the Participants), which can be solved with a Simulcast from the previously active speaker to the RTP switch.

3.3. Receiver Adaptation in Multicast/Broadcast

When using Broadcast or Multicast technology to distribute real-time media streams to large populations of receivers there can still be significant heterogeneity among the receiver population. This can depend on several factors:

Network Bandwidth: The network paths to individual receivers will have variations in the bandwidth. Thus putting different limits on the supported bit-rates that can be received.

Endpoint Capabilities: The endpoint's hardware and software can have varying capabilities in relation to screen resolution, decoding capabilities, and supported media codecs.

To handle these variations, a transmitter of real-time media may want to apply Simulcast to its Source Packet Streams and provide a set of media configurations, enabling the receivers to select the best fit from these sets themselves. The endpoint capabilities will usually result in a single initial choice. However, the network bandwidth can vary over time, which requires a client to continuously monitor its reception to determine if the received media streams still fit within the available bandwidth. If not, another Simulcast media configuration containing a thinner set of Source Packet Streams will have to be chosen.

When one uses IP multicast, the level of Simulcast granularity that the receiver can select from is by choosing different multicast addresses. Thus, different Simulcast versions need to be put on different Media Transports using different multicast addresses. If these Simulcast versions are described using SDP, they need to be part of different SDP media descriptions, as SDP binds to transport on media description level. To enable more than the initial choice to function well, there is a need to enable correct mapping of Source Packet Streams in one Simulcast media configuration to a corresponding Source Packet Stream in another Simulcast media configuration on another multicast group.

3.4. Receiver Media Source Preferences

The application logic that controls the Communication Session may allow receiving Participants to apply preferences to the characteristics of the Source Packet Stream they receive, for example in terms of the aspects listed in Section 3.1. Sending a Simulcast of Source Packet Streams is one way of accommodating receivers with conflicting or otherwise incompatible preferences.

4. Requirements

The following requirements need to be met to support the use cases in previous sections:

REQ-1: Identification. It must be possible to identify a set of simulcasted Source Packet Streams as originating from the same Media Source:

REQ-1.1: In SDP signaling.

REQ-1.2: On RTP/RTCP level.

REQ-2: Transport usage. The solution must work when distributing different Simulcast versions on:

REQ-2.1: Same Media Transport and RTP session.

REQ-2.2: Different Media Transports and RTP sessions.

REQ-3: Capability negotiation. It must be possible that:

REQ-3.1: Sender can express capability of sending simulcast.

REQ-3.2: Receiver can express capability of receiving simulcast.

REQ-3.3: Sender can express maximum number of Simulcast versions that can be provided.

REQ-3.4: Receiver can express maximum number of Simulcast versions that can be received.

REQ-3.5: Sender can detail the characteristics of the Simulcast versions that can be provided.

REQ-3.6: Receiver can detail the characteristics of the Simulcast versions that it prefers to receive.

REQ-4: Distinguishing features. It must be possible to have different Simulcast versions use different values for any combination of:

REQ-4.1: Codec. This includes both codec type and configuration options for both codec and RTP packetization. It also includes different layers from a scalable codec, but only as long as those layers are possible to identify on RTP level.

REQ-4.2: Bitrate of Source Packet Stream.

REQ-4.3: Sampling in spatial as well as in temporal domain.

REQ-5: Compatibility. It must be possible to use Simulcast in combination with other RTP mechanisms that generate additional Source Packet Streams:

REQ-5.1: RTP Retransmission [RFC4588].

REQ-5.2: RTP Forward Error Correction [RFC5109].

REQ-6: Interoperability. The solution must also be able to use in:

REQ-6.1: Interworking with non-simulcast legacy clients using a single Media Source per media type.

REQ-6.2: WebRTC "Unified Plan" environment.

5. Proposed Solution Overview

Signaling Simulcast is about negotiating between media sender and receiver what the different Simulcast versions should be, how to identify them in terms of Source Packet Streams, and how to inter-relate those Source Packet Streams.

The proposed solution consists of:

- o Signaling Simulcast capability in an optional, pre-stage Offer/Answer:
 - * Separate send and receive Simulcast capabilities as SDP session level attributes.
 - * Media properties that are supported as base for different Simulcast versions are listed as parameters that are also possible to rank.
 - * Early indication of maximum number of available encoding/decoding resources on SDP media level.
- o Including detailed information for the Simulcast in a main Offer/Answer:
 - * Including Simulcast capability indications, as described above, being kept from the pre-stage Offer/Answer, if any.
 - * Defining and labeling of the media configuration for each Simulcast version to be sent or received.

- * The media configuration for a Simulcast version can include acceptable parameter ranges for parameters that are most likely used to distinguish Simulcast versions.
 - * Indicating the use of Simulcast, separately per direction, by grouping the defined media configurations, not individual streams, that will constitute the Simulcast.
 - * Allowing that any one of the media configurations in a specific Simulcast is signaled inactive from the start of the session. This is defined as equivalent to the affected Source Packet Stream being in PAUSED state [I-D.westerlund-avtext-rtp-stream-pause].
 - * Adding and/or modifying SDP media descriptions as needed to accommodate the negotiated Simulcast streams.
 - * Parameter limits to the aggregate of media configurations are signaled by existing SDP attributes on session and media description level.
 - * Including media level indication of maximum number of available encoding/decoding resources on SDP media level. They MAY be modified compared to the pre-stage Offer/Answer, if any.
 - * Identifying which Source Packet Stream corresponds to which media configuration by including the configuration label as part of the SDES item SRCNAME [I-D.westerlund-avtext-rtcp-sdes-srcname] information include in the RTP and RTCP packets. The optional mechanism for source specific signalling defined in SRCNAME could be used to let Simulcast sender pre-announce such a relationship before sending the Source Packet Stream.
- o Adding Simulcast information to the Source Packet Stream:
- * Identifying Source Packet Streams from same Media Source using the new RTCP SDES Item SRCNAME [I-D.westerlund-avtext-rtcp-sdes-srcname], and as described there including the possibility to send the same information as an RTP Header Extension [RFC5285].
 - * Using PAUSE/RESUME [I-D.westerlund-avtext-rtp-stream-pause] functionality to temporarily turn individual Simulcast versions on or off.

6. Proposed Signaling

This section further details the signaling solution outlined above (Section 5).

6.1. Simulcast Capability

There are numerous media properties that can be varied to construct a set of Simulcast versions. A Simulcast enabled endpoint could also support Simulcast based on several of those properties. As long as those properties are relatively independent and if each Simulcast version need explicit definition in the SDP, this would lead to an exponential number of Simulcast version candidates and a very long SDP that is likely also hard to interpret. There is thus a need to limit the Simulcast version candidates included in the SDP to cover as small set of properties as possible.

If a legacy endpoint not supporting Simulcast were to be presented with an SDP including media descriptions for a set of Simulcast versions, it may not know how to correctly handle or interpret these "surplus" media descriptions.

Based on the functionality that Simulcast is intended to achieve, it should be clear that the reasons to send Simulcast versions are not the same as to receive Simulcast versions, seen from a single endpoint.

For these reasons, it is proposed to define two new SDP session level attributes, "a=sim-send-cap" and "a=sim-recv-cap", which explicitly signal support for Simulcast media transmission and Simulcast media reception, respectively, for that media description. "a=sim-send-cap" and "a=sim-recv-cap" MAY be used independently and simultaneously. These attributes are also proposed to have parameters indicating the media properties used to create the Simulcast versions, and their preferred ranking. The meaning of the attributes on SDP media level is undefined and MUST NOT be used.

```

simulcast-cap    = "a="( "sim-send-cap:" / "sim-recv-cap:" )
                  cap-prop-list
cap-prop-list    = cap-prop-entry *(WSP cap-prop-entry)
cap-prop-entry   = cap-prop ["=" q-value]
cap-prop         = "rtpmap"
                  / "fmp"
                  / "imageattr"
                  / "framerate"
                  / token ; for future extensions
q-value          = ( "0" "." 1*2DIGIT )
                  / ( "1" "." 1*2("0") )
                  ; Values between 0.00 and 1.00
; WSP and DIGIT defined in [RFC5234]
```

; token defined in [RFC4566]

Figure 1: ABNF for Simulcast Capability

The media property values are taken from existing (and could be extended to cover other or future) SDP attributes that express media properties that can be varied to create different Simulcast versions:

rtpmap: Differences in codec type, sampling rate (see Section 4), and number of channels.

fmt: Differences in codec-specific encoding parameters.

imageattr: Differences in video resolution and aspect ratio [RFC6236].

framerate: Differences in framerate.

The optional q-value expresses the relative preference to base a Simulcast version on that media property, with 1.00 meaning maximum (100%) preference and 0.00 meaning no (0%) preference. Several media properties can share the same q-value, in which case they are equally preferred. Not including any q-value for a media property value SHALL default to a q-value of 1.00.

The list of media properties is made extensible, to allow introducing additional dimensions for Simulcast versions.

6.1.1. Declarative Use

When used as a declarative media description, sim-recv-cap indicates the configured end-point's required capability to recognize and receive a specified set of Source Packet Streams as Simulcast streams. In the same fashion, sim-send-cap requests the end-point to send a specified set of Source Packet Streams as Simulcast streams. sim-recv-cap and sim-send-cap MAY be used independently and at the same time and they need not specify the same capability properties.

6.1.2. Offer/Answer Use

An offerer wanting to use Simulcast SHALL include either one or both of those attributes, depending on in which direction(s) Simulcast is both supported and desirable. An offerer that receives an answer without "a=sim-send-cap" or "a=sim-recv-cap" MUST NOT define or use any Simulcast alternatives in that direction to the answerer.

An answerer that does not understand the concept of Simulcast will also not know those attributes and will remove them in the SDP answer, as defined in existing SDP Offer/Answer procedures. An answerer that does understand the attributes and that wants to support Simulcast in the indicated direction SHALL reverse directionality of the attribute; "sim-send-cap" becomes "sim-recv-cap" and vice versa, and include it in the answer.

An offerer that intends to send Simulcast alternatives and thus includes "a=sim-send-cap", MUST also include at least one media property parameter that it intends to use to construct the Simulcast alternatives, but it MAY include more media property parameters. Including multiple media property parameters in "a=sim-send-cap" SHALL be interpreted as an offer to send Simulcast versions covering all combinations thereof, but MAY be further restricted by other information in the SDP such as for example the number of simulcast-related media descriptions in the SDP or use of max-ssrc signaling [I-D.westerlund-mmusic-max-ssrc].

An offerer that is capable of receiving Simulcast alternatives and thus includes "a=sim-recv-cap", MUST also include at least one media property parameter that it is willing to use as discriminator between received Simulcast alternatives, but MAY include more media property parameters. Including multiple media property parameters in "a=sim-recv-cap" SHALL be interpreted as an offer to receive Simulcast versions covering all combinations thereof, but MAY be further restricted by other information in the SDP such as for example the number of simulcast-related media descriptions in the SDP or use of max-ssrc signaling [I-D.westerlund-mmusic-max-ssrc].

An answerer that either lacks the capability or does not desire to use Simulcast versions based on a certain media property parameter in a specific direction MUST remove such media property parameter from "a=sim-send-cap" or "a=sim-recv-cap". The answerer MUST NOT add any media property parameters that were not included in the offer.

An answerer SHOULD take the offerer's q-values into account when choosing which media configurations (Section 6.2) to include in the answer and how to group them (Section 6.3) into the resulting Simulcast(s).

6.2. Media Configuration

Media that constitutes a Simulcast version has certain desirable characteristics that is meant to suit one category of diverse receivers (Section 3.1). A receiver that is willing to receive Simulcast streams must be given sufficient means to express what it is capable of and desires to receive. A sender that is willing to

send Simulcast streams must similarly be given sufficient means to express what it is capable of and desires to send.

An obvious candidate to express those characteristics is the media format in an SDP media description, defined by the `rtpmap` and `fntp` attributes, which is typically mapped to an RTP Payload Type. Some of the most interesting characteristics for Simulcast purposes are however not included in `rtpmap` or `fntp`, but are instead defined as separate attributes. Some of those individual attributes are possible to directly relate to a defined media format and could form a configuration together with the media format, but some attributes cannot be related to a specific media format and using the existing media format as a common identifier for a media configuration is not fully sufficient.

The act of Simulcast is trying to handle senders and receivers belonging to the vast multi-dimensional parameter space of "media configuration" by sub-dividing that parameter space into manageable and meaningful sub-sets. Communication between a sender and a receiver can be established successfully only when the actually sent media configuration (sub-set) fits within the receiver's available media configuration sub-set. At the same time, practical and implementation aspects often limits the size of those sub-sets. When that receiver or sender sub-set is either too small or is not known, the probability of successful communication decreases significantly. To increase the probability of finding a match between sender and receiver media configurations, it is essential that a media configuration can be a set instead of a single point in the parameter space, i.e. include parameter listings and/or ranges instead of single values.

Therefore, it is proposed to define a new media level SDP attribute, "a=config-id", which has relate the needed parameter types and the corresponding value ranges that together constitute a Simulcast media configuration. Each SDP media description MAY contain zero or more config-id attributes. The meaning of the attribute on SDP session level is undefined and MUST NOT be used.

```
configuration      = "a=config-id:" config-id WSP config-dir
                    WSP config-list
config-id          = token
config-dir         = "send"
                  / "recv"
config-list        = config-entry *(WSP config-entry)
config-entry       = "pt" "=" pt-value *("," pt-value)
                  / image-attr
                  / "framerate" "=" fr-param
                  / "b" "=" bw-mod ":" bw-value *1("-" bw-value)
```

```

                                / ext-config-id [ "=" ext-config-value ]
                                ; for future ext
image-attr                     = "imageattr" "=" resolution-list
resolution-list                = resolution-set *("," resolution-set)
ext-config-id                  = token
ext-config-value               = non-ws-string
pt-value                       = 1*3DIGIT ; could be made more strict
resolution-set                 = "[" "x=" xyrange "," "y=" xyrange *key-values "]"
key-values                     = ( "," key-value )
key-value                      = ( "sar=" srange )
                                / ( "par=" prange )
                                / ( "q=" qvalue )
onetone                        = "1" / "2" / "3" / "4" / "5"
                                / "6" / "7" / "8" / "9"
xyvalue                        = onetone *5DIGIT
step                           = xyvalue
xyrange                        = ( "[" xyvalue ":" [ step ":" ] xyvalue "]" )
                                / ( "[" xyvalue 1*( "," xyvalue ) "]" )
                                / ( xyvalue )
spvalue                        = ( "0" "." onetone *3DIGIT )
                                / ( onetone "." 1*4DIGIT )
srange                         = ( "[" spvalue 1*( "," spvalue ) "]" )
                                / ( "[" spvalue "-" spvalue "]" )
                                / ( spvalue )
prange                         = ( "[" spvalue "-" spvalue "]" )
qvalue                         = ( "0" "." 1*2DIGIT )
                                / ( "1" "." 1*2("0") )
fr-param                       = fr-value *("," fr-value)
                                / fr-value "-" fr-value
fr-value                       = 1*3DIGIT [ "." 1*2DIGIT ]
bw-mod                         = "AS"
                                / "TIAS"
                                / token ; for future extensions
bw-value                       = 1*DIGIT
; WSP, DQUOTE and DIGIT defined in [RFC5234]
; token and non-ws-string defined in [RFC4566]

```

Figure 2: ABNF for Media Configuration

A media configuration is thus identified by:

config-id: A token that identifies the media configuration, which MUST be unique across all media configurations and media descriptions in the SDP.

config-dir: The direction for the stream(s) receiving the media configuration, as seen from the part issuing the SDP.

The media configuration MUST contain at least one and MAY contain more of the below media configuration entries. Each entry type MUST NOT appear more than once in every media configuration.

pt: A comma-separated list of media formats, RTP payload types, which MUST be defined within the same media description as config-id. This describes the allowed set of codecs or codec configurations for this media configuration. MUST be present in every media configuration.

imageattr: An OPTIONAL listing of preferred image resolutions for this media configuration. MUST NOT be used with other than video and image media types. An imageattr media configuration entry MUST NOT conflict with any "a=imageattr" attribute present in the same media description.

framerate: An OPTIONAL range or enumeration of preferred framerates for this media configuration. MUST NOT be used with other than video media types. The high end of the range MUST be equal to or larger than the low end. An enumerating framerate media configuration entry MUST include the value of the "a=framerate" attribute, if any. A framerate range media configuration entry MUST include the "a=framerate" value in the range.

b: An acceptable bandwidth range for this media configuration. Either one of the defined bandwidth modifiers MAY be used, which MUST share semantics with corresponding bandwidth modifiers from the SDP bandwidth attribute. The bandwidth value MUST be interpreted as defined by the bandwidth modifier. The high end of the range MUST be equal to or larger than the low end. The high end of the range MUST NOT exceed the bandwidth parameter in the same media description, if any. The sum of bandwidth range low ends for all media configurations within a media description MUST NOT exceed the value of that media description's bandwidth parameter. MUST be present in every media configuration.

Media configuration entry types "pt" and "b" MUST be supported by all implementations of this specification. Otherwise, an implementation MAY ignore any media configuration entry types that are not understood. A media configuration MAY be re-used to describe more than a single Source Packet Stream.

6.2.1. Simulcast Limitations

The Session and Media level attributes and parameters outside of individual media configurations (a=config-id) provides limitations on the set of media configurations in simultaneous use. For example a media description bandwidth limitation using b=AS would apply on all

the Packet Streams sent within the scope of that media description, thus forcing the sum of the media configuration bandwidth in use to share that available bandwidth. Don't forget other Packet Streams such as RTP retransmission or FEC flows that also needs to be included.

There exist a number of different limitations, and this section does not intend to be complete. The payload formats and their configurations can offer limitations, for example video profile and levels imposes a joint limit on bit-rate, frame-rate and resolution. The bandwidth parameters on session and media description level apply according to their semantics and their level. Packetization limitations, e.g. maxptime, as well as recommendations apply to all the configurations within the scope where this parameter is defined.

It is important to note that limits, such as bandwidth expressed within a media configuration are not limited by the media description values. First of all, the sum of bit-rates across all media configurations in a media description can be greater than the media description limit as not all configurations may be in simultaneous use. For example, only a single configuration can be enabled, which is then allowed to consume the full outer limit. Secondly, the media configuration directionality needs to be taken into account, for example that SDP receiver limitations are not applied to the sender configuration.

6.2.2. Declarative Use

When used as a declarative media description, config-id with rcv parameter indicates the configured end-point's required media configuration to receive a specified set of Source Packet Streams as Simulcast streams. In the same fashion, config-id with send parameter requests the end-point to use the specified media configuration when sending a specified set of Source Packet Streams as Simulcast streams.

6.2.3. Offer/Answer Use

An offerer wanting to use Simulcast in a specific direction SHALL use config-id to describe the media configurations to use in that direction in the Offer.

An answerer receiving a config-id media configuration for a specific direction, accepting to use that media configuration SHALL include a corresponding media configuration with the reverse direction in the Answer. The config-id identification value MUST be kept between the Offer and the Answer. An answerer not accepting to use a specific media configuration SHALL remove it from the Answer.

The Answer MUST keep exactly the same media configuration types in a media configuration as were present in the corresponding media configuration in the Offer.

The answerer MAY remove values from enumerations and MAY reduce ranges of media configuration entries in the Answer. If the reduced media configuration entry relates to the answerer's send direction, negotiation is complete and no further action is needed. If the reduced media configuration relates to the answerer's receive direction, the offerer SHOULD send another Offer where that related, send direction media configuration is reduced at least to the level in the previous Answer, but MAY be reduced even more, and MAY be removed entirely.

6.3. Grouping Simulcast Configurations

A set of media configurations (Section 6.2) is needed to describe a Simulcast. Each Source Packet Stream in the Simulcast share the same Media Source, but have different media configurations. Thus, the actual grouping of media configurations is what defines a specific Simulcast. It is proposed to define two new media level and session level SDP attributes, "a=sim-send" and "a=sim-recv", which uses config-id values to group media configurations for the purpose of Simulcast transmission and reception, respectively. "a=sim-send" and "a=sim-recv" MAY be used independently and simultaneously. They MAY be used on session level to group media configurations when different Simulcast encodings of a Media Source are to be sent in different Media Transports and RTP sessions. They MAY also be used on media level to group media configurations when different Simulcast encodings of a Media Source are to be sent based on the same media description and thus use the same Media Transport and RTP session. When used on media level, the Simulcast direction MAY conflict with the general media description direction, but a conflict MUST be interpreted as the Simulcast being effectively inhibited. For example, sim-send in a recvonly media description means that no Simulcast Source Packet Streams are sent.

```

simulcast           = "a="( "sim-send:" / "sim-recv:" ) config-id-list
config-id-list      = config-item *(WSP config-item)
config-item         = config-id [":" config-param-list]
config-id           = token
config-param-list   = config-param *("," config-param)
config-param        = "inactive"
                   / token ["=" param-value] ; for future extension
param-value         = 1*(value-char)
value-char          = DQUOTE non_ws_string DQUOTE
                   / token-char / %x28 / %x29 / %x2F / %x3A-3C
                   / %x3E-40 / %x5B-5D ; VCHAR except "=" and ","

```

```
; WSP and VCHAR defined in [RFC5234]
; token, token-char and non_ws_string defined in [RFC4566]
```

Figure 3: ABNF for Simulcast Configuration Grouping

The config-id identification of a media configuration MUST be defined by a "config-id" attribute in any of the media descriptions that are part of the SDP.

6.3.1. Declarative Use

When used as a declarative media description, sim-recv indicates the configured end-point's required ability to receive Source Packet Streams with the specified set of media configurations as Simulcast streams. In the same fashion, sim-send requests the end-point to send Source Packet Streams with the specified set of media configurations as Simulcast streams.

The configuration parameter "inactive" SHALL be interpreted as the related Source Packet Stream is in PAUSED state [I-D.westerlund-avtext-rtp-stream-pause] at the start of the session, and applicable RTP level procedures from that specification SHALL be applied.

6.3.2. Offer/Answer Use

An offerer wanting to send a set of Source Packet Streams as Simulcast streams includes sim-send in the Offer to describe which media configurations to use for that Simulcast. Similarly, an offerer wanting to receive a set of Source Packet Streams as Simulcast streams includes sim-recv in the Offer to describe which media configurations to use for that Simulcast.

An answerer receiving sim-send, accepting to receive those media configurations as Simulcasted Source Packet Streams SHALL include sim-recv with the accepted media configurations in the Answer. Similarly, an answerer receiving sim-recv, accepting to send those media configurations as Simulcasted Source Packet Streams SHALL include sim-send with the accepted media configurations in the Answer. An answerer MAY remove media configurations from sim-send or sim-recv included in the Answer compared to the ones included in the sim-send or sim-recv in the Offer. The answerer MUST NOT add any media configurations to sim-send or sim-recv in the Answer that were not in the corresponding ones in the Offer.

An "inactive" parameter present in the Offer MUST be kept in the Answer. The Answer MAY add an "inactive" parameter to any of the

media configurations. An "inactive" parameter on a media configuration in "sim-recv" is equivalent to a PAUSE (or in some cases, an equivalent TMMBR 0) message [I-D.westerlund-avtext-rtp-stream-pause] being sent for the received Source Packet Stream at the start of the session, and applicable RTP level procedures from that specification SHALL be applied. An "inactive" parameter on a media configuration in "sim-send" is equivalent to the related Source Packet Stream being in PAUSED state at the start of the session, and applicable RTP level procedures SHALL be applied.

The number of different Source Packet Streams used for a Simulcast related to a single media description MUST NOT exceed the number of listed media configurations in the corresponding sim-recv in that media description sent by the media receiver.

6.4. Relating Simulcast Versions

To ensure that Simulcast Packet Streams can be related correctly on RTP level, SDES SRCNAME [I-D.westerlund-avtext-rtcp-sdes-srname] MUST be used to label Simulcast versions belonging to the same Media Source. The RTP Header Extension option of that specification MAY be used with Simulcast.

The SRCNAME identifier for Simulcast MUST contain a first part that uniquely identifies the Media Source within a given CNAME, followed by a single "." (period) and the config-id as defined above (Section 6.2).

The SRCNAME parameter to source-specific signaling [RFC5576] ("a=ssrc") MAY be used for Source Packet Streams in the send direction to relate SRCNAME to SSRC already in the SDP.

6.5. Two-Phase Negotiation

The new "a=sim-send-cap" and "a=sim-recv-cap" attributes MAY be included in the SDP as an optional pre-stage in a two-phased approach, where the pre-stage involves a first SDP Offer/Answer procedure that only establishes Simulcast capability at both the offerer and the answerer. This has the additional advantage to avoid sending media descriptions related to Simulcast to an endpoint that does not support simulcast. In case two Offer/Answer procedures are already used for other reasons, it will not incur any significant extra signaling round-trips. Such other two-phase techniques include use of SIP OPTIONS, SIP UPDATE [RFC3311] with reliable provisional responses, and BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation].

Thus, when using the pre-stage Offer/Answer, it SHOULD NOT include any simulcast-grouped media descriptions, which SHOULD then instead be added in a main Offer/Answer phase. When using the pre-stage Offer/Answer, half a signaling round-trip time can sometimes be saved if main phase is initiated by the Simulcast receiver, meaning that the endpoint that included "a=sim-recv" in the pre-stage SDP is the offerer in the main phase. If both endpoints are Simulcast receivers, it does not matter which endpoint sends the main Offer, using regular Offer/Answer rules to handle any race conditions.

It is not possible to use any pre-stage to establish capability with declarative SDP, in which case it SHALL be by-passed, using only the main phase directly.

6.6. Signaling Examples

These examples are for a case of client to video conference service using a centralized media topology with an RTP mixer.

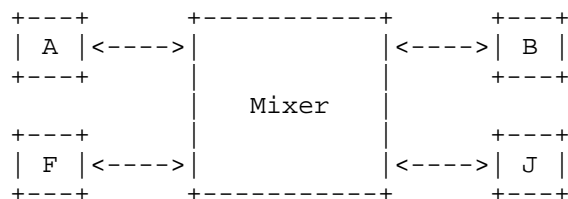


Figure 4: Four-party Mixer-based Conference

6.6.1. Unified Plan Client

Alice is calling in to the mixer with a Simulcast-enabled Unified Plan client capable of a single Media Source per media type. The only difference to a non-Simulcast client is capability to send video resolution [RFC6236] ("imageattr") and framerate based Simulcast. Alice uses a pre-stage Offer, which looks like:

```

v=0
o=alice 2362969037 2362969040 IN IP4 192.0.2.156
s=Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.156
b=AS:665
a=sim-send-cap:imageattr framerate
m=audio 49200 RTP/AVP 96 8
b=AS:145
a=rtpmap:96 G719/48000/2
a=rtpmap:8 PCMA/8000
  
```

```
m=video 49300 RTP/AVP 97
b=AS:520
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=imageattr:97 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
```

Figure 5: Unified Plan Simulcast Pre-Stage Offer

In this pre-stage, the only thing in the SDP that indicates Simulcast capability is the line in the video media description containing the "sim-send-cap" attribute, which also indicates that sent Simulcast versions can differ in video resolution and/or framerate.

The Answer from the server indicates both that it too is Simulcast capable and that it would prefer to use video resolution ("imageattr") based Simulcast, but that it supports both video resolution and framerate. Should it not have been Simulcast capable, the "a=sim-recv-cap" line would not have been present and communication would have started with the media negotiated in the SDP.

```
v=0
o=server 823479283 1209384938 IN IP4 192.0.2.2
s=Answer to Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.43
b=AS:665
a=sim-recv-cap:imageattr=1.0 framerate=0.8
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 97
b=AS:520
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=imageattr:97 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
```

Figure 6: Unified Plan Simulcast Pre-Stage Answer

Since the server is the Simulcast media receiver, it immediately initiates another Offer/Answer including details on the Simulcast versions. The server also keeps the "sim-recv-cap" as explicit Simulcast capability indication in this main Offer/Answer. Note that the "non-simulcast" media can be started already now, before the main

Offer/Answer, with the only restriction that the Simulcast functionality is not yet established.

```
v=0
o=server 823479283 1209384938 IN IP4 192.0.2.2
s=Server Inviting Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.43
b=AS:825
a=sim-recv-cap:imageattr=1.0 framerate=0.8
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 97
b=AS:2200
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=config-id:a recv pt=97 imageattr=[x=640,y=360],[x=1280,y=720] \
    framerate=25-60 b=AS:500-2500
a=config-id:b recv pt=97 imageattr=[x=320,y=180],[x=640,y=360] \
    framerate=25-60 b=AS:150-500
a=config-id:c recv pt=97 imageattr=[x=256,y=144],[x=320,y=180] \
    framerate=10-30 b=AS:100-250
a=sim-recv:a b c
```

Figure 7: Unified Plan Simulcast Main Offer

The server chooses to structure the Answer according to Unified Plan and has added three config-id lines in the video media description, one for each Simulcast media configuration that it is prepared to receive. Each media configuration refers to a defined media format, and lists a set of preferred video resolutions as well as a range of acceptable framerates, concluded by a bandwidth range. It also includes the sim-recv attribute for those three media configurations, indicating that the Simulcast it is prepared to receive in this media description can include one or more of those media configurations.

Alice's Answer is:

```
v=0
o=alice 2362969037 2362969040 IN IP4 192.0.2.156
s=Final answer from Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.156
b=AS:825
a=sim-send-cap:imageattr framerate
m=audio 49200 RTP/AVP 96
```



```

b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 97
b=AS:520
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=config-id:b send pt=97 imageattr=[x=640,y=360] \
    framerate=25-30 b=AS:150-400
a=config-id:c send pt=97 imageattr=[x=320,y=180] \
    framerate=10-12.5 b=AS:100-150
a=sim-send:b c:inactive
a=ssrc:31053821 cname=SDIe93850aQFid9P srcname=l.b
a=ssrc:43298172 cname=SDIe93850aQFid9P srcname=l.c
a=imageattr:97 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]

```

Figure 8: Unified Plan Simulcast Main Answer

The Simulcast capability, `sim-send-cap`, is kept from Alice's previous Offer. One of the media configurations from the server Offer, `config-id:a`, is not acceptable to Alice's client for some reason and is removed from the Answer. The resulting Simulcast, described by `sim-send`, thus contains two media configurations, `b` and `c`, where `c` is initially set to "inactive" that effectively means it is paused from the start of the session. The media configuration parameter value ranges are in some cases reduced, which makes a more precise definition of what will actually be sent. This Answer SDP also includes a specification of the SSRC values that will be sent and what media configurations those SSRC will carry, by including the `srcname` parameter. The first part of `srcname`, before the ".", is the Media Source identification. Both SSRC share the same Media Source identification, since they are part of the same Simulcast. The second part, after the ".", is the `config-id` of the media configuration sent with that SSRC.

6.6.2. Multi-Transport Client

Bob is calling in to the mixer with a Simulcast-enabled client, like Alice's capable of a single Media Source per media type, but also capable of sending Source Packet Streams as Simulcast versions on separate Media Transports. In this example, Bob's client knows that the server is capable of Simulcast and does not use any pre-stage Offer, but goes straight to the main Offer.

```

v=0
o=bob 94572932847 3429478298 IN IP4 192.0.2.93
s=Offer from Simulcast Enabled Multi-Transport Client

```

```

t=0 0
c=IN IP4 192.0.2.93
b=AS:825
a=sim-send-cap:imageattr=1.0 framerate=0.9
a=sim-send:x y
m=audio 50138 RTP/AVP 101
b=AS:145
a=rtpmap:101 G719/48000/2
m=video 50226 RTP/AVP 118
b=AS:500
a=rtpmap:118 H264/90000
a=fmtp:118 profile-level-id=42c01e
a=config-id:x send pt=118 imageattr=[x=320,y=180],[x=640,y=360] \
    framerate=25-50 b=AS:200-500
a=ssrc:3929384298 cname=Nsdko39Oen828FKn srcname=M.x
a=imageattr:118 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
m=video 50228 RTP/AVP 119
b=AS:150
a=config-id:y send pt=119 imageattr=[x=256,y=144],[x=320,y=180] \
    framerate=12.5-25 b=AS:100-200
a=ssrc:1923419284 cname=Nsdko39Oen828FKn srcname=M.y
a=imageattr:119 send [x=320,y=180] [x=256,y=144]
a=sendonly

```

Figure 9: Multi-Transport Simulcast Main Offer

As can be seen from above, this Offer uses `sim-send` on session level and has split the Simulcast media configurations on two media descriptions, in order to be able to use separate Media Transports and enable differentiated treatment of the two Simulcast streams.

The server accepts this structure to the Answer:

```

v=0
o=server 283479882 9384298374 IN IP4 192.0.2.2
s=Server Answering Simulcast Enabled Multi-Transport Client
t=0 0
c=IN IP4 192.0.2.45
b=AS:825
a=sim-recv-cap:imageattr framerate
a=sim-recv:x y
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 118
b=AS:500

```

```

a=rtpmap:118 H264/90000
a=fmtp:118 profile-level-id=42c01e
a=config-id:x recv pt=118 imageattr=[x=640,y=360] \
    framerate=25-50 b=AS:350-500
a=imageattr:118 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
m=video 49300 RTP/AVP 119
b=AS:150
a=rtpmap:119 H264/90000
a=fmtp:119 profile-level-id=42c01e
a=config-id:y recv pt=119 imageattr=[x=256,y=144] \
    framerate=12.5-25 b=AS:120-150
a=imageattr:119 recv [x=320,y=180] [x=256,y=144]
a=recvonly

```

Figure 10: Multi-Transport Simulcast Main Answer

6.6.3. Multi-Source Client

Fred is calling in to the same conference as in the examples above with a three-camera, three-display system, thus capable of handling three separate Media Sources in each direction, where each Media Source is also Simulcast-enabled in the send direction. Fred's client is a Unified Plan client, restricted to a single Media Source per media description.

```

v=0
o=fred 238947129 823479223 IN IP4 192.0.2.125
s=Offer from Simulcast Enabled Multi-Source Client
t=0 0
c=IN IP4 192.0.2.125
b=AS:825
a=sim-send-cap:imageattr=1.0 framerate=0.5

m=audio 49200 RTP/AVP 98
b=AS:145
a=rtpmap:98 G719/48000/2

m=video 49600 RTP/AVP 100
b=AS:3500
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42c02a
a=config-id:lh send pt=100 imageattr=[x=1920,y=1080] \
    framerate=30-60 b=AS:2000-3500
a=config-id:lm send pt=100 imageattr=[x=1280,y=720] \
    framerate=15-60 b=AS:1000-2000
a=config-id:ll send pt=100 imageattr=[x=640,y=360] \

```

```
    framerate=10-60 b=AS:200-1000
a=sim-send:1h 1m 1l
a=ssrc:2397234521 cname=EkeS32892FeO29DK srcname=1.1h
a=ssrc:1023894789 cname=EkeS32892FeO29DK srcname=1.1m
a=ssrc:4029284928 cname=EkeS32892FeO29DK srcname=1.1l
a=imageattr:100 send [x=1920,y=1080] [x=1280,y=720] [x=640,y=360] \
    recv [x=1920,y=1080] [x=1280,y=720] [x=640,y=360]

m=video 49600 RTP/AVP 100
b=AS:3500
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42c02a
a=config-id:2h send pt=100 imageattr=[x=1920,y=1080] \
    framerate=30-60 b=AS:2000-3500
a=config-id:2m send pt=100 imageattr=[x=1280,y=720] \
    framerate=15-60 b=AS:1000-2000
a=config-id:2l send pt=100 imageattr=[x=640,y=360] \
    framerate=10-60 b=AS:200-1000
a=sim-send:2h 2m 2l
a=ssrc:2301017618 cname=EkeS32892FeO29DK srcname=2.2h
a=ssrc:639711316 cname=EkeS32892FeO29DK srcname=2.2m
a=ssrc:3293473905 cname=EkeS32892FeO29DK srcname=2.2l
a=imageattr:100 send [x=1920,y=1080] [x=1280,y=720] [x=640,y=360] \
    recv [x=1920,y=1080] [x=1280,y=720] [x=640,y=360]

m=video 49600 RTP/AVP 100
b=AS:3500
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42c02a
a=config-id:3h send pt=100 imageattr=[x=1920,y=1080] \
    framerate=30-60 b=AS:2000-3500
a=config-id:3m send pt=100 imageattr=[x=1280,y=720] \
    framerate=15-60 b=AS:1000-2000
a=config-id:3l send pt=100 imageattr=[x=640,y=360] \
    framerate=10-60 b=AS:200-1000
a=sim-send:3h 3m 3l
a=ssrc:4115355057 cname=EkeS32892FeO29DK srcname=3.3h
a=ssrc:3196538337 cname=EkeS32892FeO29DK srcname=3.3m
a=ssrc:3757973912 cname=EkeS32892FeO29DK srcname=3.3l
a=imageattr:100 send [x=1920,y=1080] [x=1280,y=720] [x=640,y=360] \
    recv [x=1920,y=1080] [x=1280,y=720] [x=640,y=360]
```

Figure 11: Fred's Multi-Source Simulcast Main Offer

The three media descriptions for video are essentially the same, except values that needs to be unique are provided unique values. The above also assumes that BUNDLE will be used across these three video media description to create a common RTP session.

7. Network Aspects

Simulcast is in defined as the act of sending multiple alternative encodings of the same underlying media source. When transmitting multiple independent streams that originate from the same source, it could potentially be done in several different ways using RTP. A general discussion on considerations for use of the different RTP multiplexing alternatives can be found in Guidelines for Multiplexing in RTP [I-D.ietf-avtcore-multiplex-guidelines]. Discussion and clarification on how to handle multiple streams in an RTP session can be found in [I-D.ietf-avtcore-rtp-multi-stream].

The network aspects that are relevant for Simulcast are:

Quality of Service: When using Simulcast it might be of interest to prioritize a particular Simulcast version, rather than applying equal treatment to all versions. For example, lower bit-rate versions may be prioritized over higher bit-rate versions to minimize congestion or packet losses in the low bit-rate versions. Thus, there is a benefit to use a Simulcast solution that supports QoS as good as possible. By separating Simulcast versions into different RTP sessions and send those RTP sessions over different Media Transports, a Simulcast version can be prioritized by existing flow based QoS mechanisms. When using unicast, QoS mechanisms based on individual packet marking are also feasible, which do not require separation of Simulcast versions into different RTP sessions to apply different QoS.

NAT/FW Traversal: Using multiple RTP sessions will incur more cost for NAT/FW traversal unless they can re-use the same transport flow, which can be achieved by either one of multiplexing multiple RTP sessions on a single lower layer transport [I-D.westerlund-avtcore-transport-multiplexing] or Multiplexing Negotiation Using SDP Port Numbers [I-D.ietf-mmusic-sdp-bundle-negotiation]. If flow based QoS with any differentiation is desirable, the cost for additional transport flows is likely necessary.

Multicast: Multiple RTP sessions will be required to enable combining Simulcast with multicast. Different Simulcast versions have to be separated to different multicast groups to allow a multicast receiver to pick the version it wants, rather than receive all of them. In this case, the only reasonable

implementation is to use different RTP sessions for each multicast group so that reporting and other RTCP functions operate as intended.

8. IANA Considerations

This document requests that five new attributes, `sim-send-cap`, `sim-recv-cap`, `sim-send`, `sim-recv`, and `config-id`. It is also requested to make a new registry of defined parameters taken from existing SDP attributes for `sim-send-cap`, `sim-recv-cap`, and `config-id`.

Formal registrations to be written.

9. Security Considerations

The Simulcast capability and configuration attributes and parameters are vulnerable to attacks in signaling.

A false inclusion of Simulcast attributes may result in generation of a second phase SDP that potentially contains a large number of non-supported media descriptions expressing Simulcast alternatives. A correct SDP implementation will however be able to reject any non-supported media descriptions and the effect from that should be limited.

A hostile removal of the Simulcast attributes will result in skipping any second phase Offer/Answer and that Simulcast is not used.

The Simulcast grouping semantics are vulnerable to attacks in the signalling. Changing the set of media configurations that are used in a Simulcast will impact the number of Source Packet Streams.

A hostile removal of Simulcast grouping will prevent streams from being interpreted as Simulcast, which obviously prevents use of the Simulcast functionality. It will also risk that intended Simulcast streams are instead presented as separate, independent streams to a receiver.

Neither of the above will likely have any major consequences and can be mitigated by signaling that is at least integrity and source authenticated to prevent an attacker to change it.

10. Contributors

Morgan Lindqvist and Fredrik Jansson, both from Ericsson, have contributed with important material to the first versions of this document.

11. Acknowledgements

12. References

12.1. Normative References

- [I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., "RTCP Source Description Item SRCNAME to Label Individual Media Sources", draft-westerlund-avtext-rtcp-sdes-srcname-03 (work in progress), October 2013.
- [I-D.westerlund-avtext-rtp-stream-pause]
Akram, A., Burman, B., Grondal, D., and M. Westerlund, "RTP Media Stream Pause and Resume", draft-westerlund-avtext-rtp-stream-pause-03 (work in progress), October 2012.
- [I-D.westerlund-mmusic-max-ssrc]
Holmberg, C., Westerlund, M., and F. Jansson, "Multiple Synchronization Sources (SSRC) in SDP Media Descriptions", draft-westerlund-mmusic-max-ssrc-02 (work in progress), September 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

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

12.2. Informative References

- [I-D.ietf-avtcore-multiplex-guidelines]
Westerlund, M., Perkins, C., and H. Alvestrand,
"Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams", draft-ietf-avtcore-multiplex-guidelines-01 (work in progress), July 2013.
- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-01 (work in progress), July 2013.
- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-00 (work in progress), April 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-05 (work in progress), October 2013.
- [I-D.lennox-raiarea-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro,
"A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-lennox-raiarea-rtp-grouping-taxonomy-03 (work in progress), October 2013.
- [I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-06 (work in progress), August 2013.

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3569] Bhattacharyya, S., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, 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.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.

Appendix A. Discussion on Receiver Diversity

Receiver diversity can be handled in a number of different ways, each with its own advantages and disadvantages. In that, there are relations between RTP Mixer processing requirement, bandwidth usage on uplink from sending Participant to RTP Mixer, bandwidth usage on downlink from RTP Mixer to receiving Participant, and media Quality of Experience at the receiving Participant.

The following is a listing of possible approaches:

1. Lowest Common Denominator: Create a single Source Packet Stream per Media Source and, assuming that everyone can receive a "simple" stream, adapt the characteristics of that Source Packet Stream already at the sending Participant to the lowest common denominator among all receiving Participants. Let the RTP Mixer forward this single Source Packet Stream to all receiving Participants. The advantages are low bandwidth usage on both uplink and downlink and low RTP Mixer processing requirements. The disadvantage is that the least capable receiver and/or network path dictates the (low) QoE for everyone else.
2. Individual Transcoding: Create a single Source Packet Stream per Media Source with characteristics governed by resources available to the sending Participant and the network path to the RTP Mixer.

Let the RTP Mixer transcode (decode and re-encode) that into individual Source Packet Streams for each receiving Participant, governed by the RTP Mixer resources, receiving Participant resources, and the network path to that Participant. The advantages are adapted although overall slightly lowered QoE (due to transcoding) to each Participant and optimised bandwidth usage on both uplink and downlink. The disadvantage is (very) high RTP Mixer processing requirements.

3. Individual Simulcast: Create individual Source Packet Streams of each Media Source to each receiving Participant, constituting a complete individual Simulcast. Let the RTP Mixer forward each individual Source Packet Stream to the targeted receiving Participant. The advantages are low RTP Mixer processing and optimised downlink bandwidth. The disadvantage is (very) high uplink bandwidth.
4. Grouped Simulcast: For each Media Source, create a "suitable" logical grouping of receiving Participants in sub-groups with respect to available receiver resources, for example the resources listed above (Section 3.1). Create a set of Source Packet Streams for this Media Source with well-chosen characteristics, where each Source Packet Stream in the set is a good-enough fit to the receiving sub-group of Participants. This set of Source Packet Streams constitutes a Simulcast of the Media Source. The size of the set and the characteristics of each Source Packet Stream can be adjusted to cater for various restrictions in the sending Participant, receiving Participants in the sub-group, and network path(s) to the Participants in the sub-group. Let the RTP Mixer forward the same Source Packet Stream to all Participants in a sub-group, for all Source Packet Streams and sub-groups. The advantages are low RTP Mixer processing, near optimum QoE, and near optimum downlink bandwidth. The disadvantages are high uplink bandwidth and arguably that downlink bandwidth and QoE are optimum only for a sub-group and not per individual receiving Participant.

A summary of the advantages and disadvantages of the above four principle alternatives is given below (Table 1):

Method	Mixer CPU	Uplink	Downlink	QoE
1	Low	Low	Low	Low
2	Very high	Optimum	Optimum	Near optimum
3	Low	Very high	Optimum	Optimum
4	Low	High	Near optimum	Near optimum

Table 1: Receiver Diversity Handling Comparison

The authors of this document believes that alternative 4, the Grouped Simulcast, can be a good tradeoff whenever supported by sufficient uplink resources.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com