

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

R. Ejzak  
Alcatel-Lucent  
February 26, 2013

Alternatives to BUNDLE  
draft-ejzak-mmusic-bundle-alternatives-01

Abstract

This paper discusses some potential modifications to the BUNDLE proposal for multiplexing of multiple media types within a single 5-tuple to address limitations of the current proposal and alternatives already considered by MMUSIC.

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 30, 2013.

Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	3
2. Issues with BUNDLE . . . . .	3
3. Potential solutions . . . . .	4
3.1. Unspecified address for subsequent m lines in SDP offer . .	4
3.2. Unspecified address for subsequent m lines in SDP answer . . . . .	6
3.3. Hybrid approach . . . . .	7
4. Discussion . . . . .	8
5. IANA Considerations . . . . .	8
6. Security Considerations . . . . .	8
7. Informative References . . . . .	8
Author's Address . . . . .	9

## 1. Introduction

BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] provides for the multiplexing of the media associated with multiple SDP [RFC4566] m lines into a single 5-tuple and RTP [RFC3550] session, thus providing many potential advantages in reducing the messages needed for ICE [RFC5245] candidate gathering (particularly for server reflexive candidates) and reducing the messaging for DTLS [RFC6347] key exchange. BUNDLE is signaled in an SDP offer by using the grouping framework to identify those m lines that are to share a single 5-tuple. The grouped m lines are all signaled with different port numbers in the first SDP offer/answer exchange to allow for successful negotiation without BUNDLE. This is a change from the previous version of BUNDLE, which signaled the same port for each bundled m line. The change was needed to work with legacy intermediaries that would fail the call on receipt an SDP offer with the same port on multiple m lines. In the event that BUNDLE negotiation succeeds, each subsequent SDP offer is sent with the same port number for each valid m line in the bundle. This is done to clearly signal to intermediaries the 5-tuple in use for each m line.

## 2. Issues with BUNDLE

BUNDLE always requires at least two SDP offer/answer exchanges to negotiate the (preferred) use of bundled media, but only one offer/answer exchange to reject bundled media. BUNDLE is intended to minimize signaling rather than to increase it - and particularly the more expensive out-of-band signaling.

Since the answerer can reject individual m lines (including the first one), it's not clear which 5-tuple will be in use for the bundle until the second offer/answer exchange completes, delaying call setup time in this case.

In the event that a subsequent SDP offer/answer exchange is needed later during the session, BUNDLE requires the signaling of the same port for each bundled m line. If there is an intermediary in the session performing 3pcc procedures as a B2BUA, then the intermediary may send an empty (SDP-less) re-INVITE request to a WebRTC endpoint to trigger sending of an SDP offer, which is a common 3pcc scenario. If the purpose of the empty re-INVITE is to establish a connection to a different media endpoint and the SDP offer is forwarded to the new endpoint via a legacy intermediary that requires unique ports on m lines, then the 3pcc procedure will fail and there is no simple work-around. While it is possible to provide an API option to request a new offer with different port numbers, there is no reliable way to anticipate when such a 3pcc scenario might occur so there is still

the risk of 3pcc scenario failure.

### 3. Potential solutions

This paper proposes two modifications to BUNDLE for consideration. As with the current version of BUNDLE, these proposals assume the signaling of different port numbers for each m line in the SDP offer to avoid call failure and retry. Since the proposals address different scenarios and are compatible with one another, both could be adopted, as also discussed below as a "hybrid" option.

Note that the paper only considers manipulation of the connection and port information in the subsequent m lines of the SDP offer and/or answer (in particular the use of the unspecified address). Zero port is not considered due to its very specific meaning on an SDP m line. There are also some restrictions on the handling of DTLS crypto information since this is shared among the bundled m lines. Changes to bandwidth, directionality, or other attributes are not considered since they are all needed to signal characteristics of the media associated with the m line.

This paper does not discuss the syntax of the unspecified address for IPv6 as this has been covered elsewhere.

#### 3.1. Unspecified address for subsequent m lines in SDP offer

This approach borrows a mechanism from Trickle ICE to avoid the signaling of any candidates for the subsequent m lines in the SDP offer. The middle box will not allocate resources for these m lines when forwarding the SDP offer, and the SDP answerer will usually respond with the unspecified address for these m lines. Even if the SDP answer includes valid connection information for these m lines, the middle box will still not allocate separate transport flows for them.

If the SDP answerer chooses to reject the first m line(s) in the bundle group in the SDP offer (by setting port in the m line to zero), it places the intended port, connection, candidate and DTLS crypto information for the bundle in the first valid m line of the SDP answer and includes the unspecified address for all other m lines in the bundle group. Because of this, it is understood by both endpoints that the 5-tuple connection, port and DTLS crypto information is to be based on the first valid m line in the SDP offer and the first valid m line in the SDP answer. It is possible for this information to be included in different m lines in the answer compared to the offer, but with this rule there is no ambiguity as to the parameters of the transport connection, and the intermediary only

sees this SDP exchange if it has previously forwarded an indication of support for BUNDLE.

In this relatively rare case where the answerer rejects the first (usually highest priority) m line, the offerer should send an updated offer acknowledging rejection of the initial m line(s) (with zero port), and with the transport information already in use for the bundle moved to the new first valid m line. Simple intermediaries that only look at transport information in the SDP, e.g., to open pinholes, would be confused without a second SDP offer.

For subsequent offers, the offerer will put the port, connection, candidate and DTLS crypto information in use for the bundle in the first valid m line of the new SDP offer to start the process again.

If the SDP answerer chooses to not bundle media, then the SDP offerer will either need to perform Trickle ICE, if supported, or to send another SDP offer with valid connection, candidate and port information for each m line.

The primary advantage of this approach is that it is unnecessary to allocate either any ICE candidates for the subsequent m lines or any middle box resources for these m lines.

Another advantage of this approach is that media setup completes in one SDP offer/answer exchange for the most common scenarios with BUNDLE where the first bundled m line is accepted by the answerer. The need of a second SDP offer/answer to support the less-preferred non-bundle scenario or to support the unlikely answerer's rejection of the first m line should be of less concern.

This approach also eliminates redundant (and potentially conflicting) transport information from multiple m lines in both SDP offers and answers, thus improving readability and slightly decreasing the size of the SDP messages.

It is possible that the middle box will not accept the SDP offer with an unspecified address, although RFC 3264 mandates support for this.

RFC 3264 indicates that the use of unspecified address for an m line signals that "neither RTP nor RTCP should be sent to the peer". It is understood with the BUNDLE extension defined here that this text is modified to mean that no RTP or RTCP is sent using the transport parameters defined for the media line.

### 3.2. Unspecified address for subsequent m lines in SDP answer

To further reduce the potential for unsupported signaling at middle boxes and to avoid problems with the unbundled case, the solution might still signal valid connection and port information for all m lines in the bundle group of the first SDP offer (as in the current BUNDLE draft), thus potentially causing the middle box to unnecessarily allocate resources. But if the SDP answerer decides to bundle media, then it can signal the unspecified address for the subsequent m lines in the bundle.

With this approach, the middle box recognizing the unspecified address in subsequent m lines will release extraneous resources and avoid failure due to inactivity.

If the SDP answerer chooses to reject the first m line(s) in the bundle group in the SDP offer (by setting port in the m line to zero), it places the intended port, connection, candidate and DTLS crypto information for the bundle in the first valid m line of the SDP answer and includes the unspecified address for all other m lines in the bundle group. Because of this, it is understood by both endpoints that the 5-tuple connection, port and DTLS crypto information is to be based on the first valid m line in the SDP offer and the first valid m line in the SDP answer. It is possible for this information to be included in different m lines in the answer compared to the offer, but with this rule there is no ambiguity as to the parameters of the transport connection, and the intermediary only sees this SDP exchange if it has previously forwarded an indication of support for BUNDLE.

In this relatively rare case where the answerer rejects the first (usually highest priority) m line, the offerer should send an updated offer acknowledging rejection of the initial m line(s) (with zero port), and with the transport information already in use for the bundle moved to the new first valid m line. Note in this case that the new SDP offer still includes valid port and connection information for all other valid m lines in the bundle group to be prepared for any 3pcc scenario. Simple intermediaries that only look at transport information in the SDP, e.g., to open pinholes, would be confused without a second SDP offer.

For subsequent offers, the offerer will put the port, connection, candidate and DTLS crypto information in use for the bundle in the first valid m line of the new SDP offer to start the process again. The new SDP offer still includes valid port and connection information for all other valid m lines in the bundle group.

This approach is robust in the presence of 3pcc scenarios that

forward SDP to different endpoints. This approach avoids sending the unspecified address to intermediaries that have not already indicated support for BUNDLE.

This approach also eliminates redundant (and potentially conflicting) transport information from multiple m lines in the SDP answer, thus improving readability and slightly decreasing the size of the SDP messages.

There is some small risk that the middle box will not recognize the unspecified address in the SDP answer (even though its support is mandated), but this risk is limited to the bundled case since the SDP for the unbundled case is not impacted.

This approach has the disadvantage that the offerer must allocate connection and candidate information for all m lines in the bundle even when only one set of transportation information is used in the bundle case.

### 3.3. Hybrid approach

The two approaches "unspecified address for subsequent m lines in SDP offer" (option O) and "unspecified address for subsequent m lines in SDP answer" (option A) can be combined into a hybrid approach as follows.

Option A is the default option for the initial SDP offer/answer exchange for a new session with an unknown endpoint. This minimizes potential problems with intermediaries and allows for completion of media setup with one SDP offer/answer exchange for most bundled cases and all non-bundled cases.

Option O can also be used for the initial SDP offer/answer exchange when it is known that bundle is likely to succeed and there is no concern with compatibility at intermediaries.

Option O is the default option for subsequent SDP offer/answer exchanges during a session once it is established that both endpoints and intermediaries support BUNDLE.

Option A is used for subsequent SDP offer/answer exchanges during a session if BUNDLE is not negotiated during the initial exchange or if there is any potential for a 3pcc scenario sending signaling through a new and potentially incompatible intermediary.

This hybrid approach combines the best features of both alternatives and provides considerable flexibility in fine tuning the SDP offer/answer exchange for different applications.

#### 4. Discussion

Of the approaches presented in the paper, the author prefers the hybrid modification of BUNDLE described above over either individual approach and prefers all of them over the current BUNDLE.

The hybrid approach allows completion of media and transport negotiation in one SDP offer/exchange in most cases, and most importantly when successfully negotiating the bundling of media.

The hybrid approach is consistent with 3pcc using either signaling option, thus avoiding potential 3pcc failure scenarios.

The hybrid approach minimizes redundant transport related information in the bundled m lines thus slightly reducing SDP size and processing requirements.

The hybrid approach allows for customization of the procedures to simplify common (e.g., WebRTC only) cases and to avoid allocation of transport resources when they are not needed.

There is some risk in including the unspecified address for certain m lines in the SDP. This risk can be limited to exposing the unspecified address only in the SDP answer to intermediaries that have already signaled support for this extended BUNDLE proposal in the forwarded SDP offer. Since support for the unspecified address is mandated by RFC 3264, this seems like a small risk.

#### 5. IANA Considerations

To be completed.

#### 6. Security Considerations

To be completed.

#### 7. Informative References

- [I-D.ietf-mmusic-sdp-bundle-negotiation]  
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-00 (work in progress), February 2012.
- [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.

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

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

#### Author's Address

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

Email: richard.ejzak@alcatel-lucent.com



MMUSIC WG  
Internet-Draft  
Intended status: Informational  
Expires: August 21, 2013

R. Even  
Huawei Technologies  
J. Lennox  
Vidyo  
Q. Wu  
Huawei Technologies  
February 17, 2013

Describing multiple RTP media streams in SDP  
draft-even-mmusic-multiple-streams-02.txt

Abstract

This document describes issues when describing multiple RTP streams in a single RTP session using SDP and considers the different RTP topologies that should be supported. The document looks at current solutions and provides paths toward addressing the issues.

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 21, 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	5
3. RTP topologies for CLUE . . . . .	5
4. Review of current directions in MMUSIC, AVText and AVTcore . .	7
5. Requirements from a solution . . . . .	9
6. SDP limitations and proposed solution . . . . .	10
6.1. single RTP stream . . . . .	11
6.2. One or multiple RTP streams . . . . .	11
7. Acknowledgements . . . . .	11
8. IANA Considerations . . . . .	11
9. Security Considerations . . . . .	11
10. References . . . . .	12
10.1. Normative References . . . . .	12
10.2. Informative References . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction

Communication systems can send and receive multiple RTP media streams. The streams can be multiple video streams from the same source/camera representing, for example, different resolutions (simulcast, scalable video (SVC)) or repair streams (FEC). They can be different streams from the same endpoint but from different cameras, for example a Telepresence system sending two views of the room from two different cameras. They can also be multiple streams from separate original endpoints, sent by a middlebox.

RTP [RFC3550] and [I-D.ietf-avtcore-multi-media-rtp-session] allow the multiplexing of multiple media of the same and different types (video with video and audio with video) in a single RTP session identified by a single transport address. The RTP streams are identified by their synchronization source identifiers (SSRC).

SIP offer answer [RFC3264] uses SDP [RFC4566] to negotiate RTP [RFC3550] media streams. This document discusses the capabilities and limitations of SDP when describing SSRC multiplexed streams.

When looking at the following offer

```
m=video 10000 RTP/AVP 31 32
```

```
a=rtpmap:31 H261/90000
```

```
a=rtpmap:32 MPV/90000
```

What does it mean one RTP session is offered with H.261 or MPV codecs for the same content, or one RTP session is offered with H.261 and MPV codecs each with different content?

This offer should really mean "arbitrarily many streams, with potentially different content, any of which could use either H.261 or MPV, potentially switching dynamically between them." Now how do we provide enough information in SDP to allow the receiver to get a better understanding of what the offer is.

Reading some text from RFC3264 it may look like a Media stream is defined as a single media instance

"The offer will contain zero or more media streams (each media stream is described by an "m=" line and its associated attributes)."

"In all cases, the formats in the "m=" line MUST be listed in order of preference, with the first format listed being preferred. In this case, preferred means that the recipient of the offer SHOULD use the

format with the highest preference that is acceptable to it."

"For each "m=" line in the offer, there MUST be a corresponding "m=" line in the answer. The answer MUST contain exactly the same number of "m=" lines as the offer. This allows for streams to be matched up based on their order"

Since SDP and [RFC3264] offer/answer describe RTP sessions, SDP's term "media stream" is poorly chosen. Careful reading reveals that a single SDP "media stream" can be used by arbitrarily many RTP streams. (Indeed, historically this was the case in SAP, the first usage of SDP, which was used to describe loosely-coupled RTP multicast sessions with arbitrarily many participants.)

The logic of RFC3264 about the preference does not work if you have multiple RTP streams in the same m-line unless the same preference applies to all the RTP streams. So when we look at solution we will also need to clarify the text in RFC3264 and most probably will need to have the right terminology for RTP session, media session across the different documents.

SDP [RFC4566] is used to describe the multimedia session. The basic model uses a two level hierarchy, consisting of session level and media level.

SDP support of multiplexing multiple media streams in one RTP session based on the RTP stream SSRC does not provide sufficient capabilities to allow each of the multiplexed RTP streams identified by SSRC to have unique attributes, for example different bandwidth. Furthermore, when an offer has multiple payload type in a single media level descriptor (m-line), this is identified as option to receive all this payload types multiplexed.

SDP provides a framework to define grouping relations between SDP media streams [RFC5888]. This framework specifies the grouping based on the SDP media session and not on RTP stream.

Some tools for supporting RTP stream level attributes per RTP streams as well as support for simulcast were proposed and this document will look at them. It was not a major problem so far since most endpoints are using a single audio and video stream and are using SDP media level descriptors (m-lines) to describe each of the streams. Some of the existing implementation when offering multiple payload types in a single m-line are doing a second offer/answer exchange offering only one of the payload types removing the rest in order to indicate that they can only receive one media type encoding at a time. Every change of media type requires an offer / answer exchange.

Currently both RTCweb and CLUE WGs have interest in better support for multiplexing either multiple RTP media streams from the same type or different types. The work in [draft-ietf-mmusic-sdp-bundle-negotiation-01] and [draft-holmberg-mmusic-sdp-mmt-negotiation-00] provides two different directions for initial bundling support options for SDP negotiation of multiplexing different media types but the problem of identifying different RTP streams with different attributes is still not fully solved. There is a dependency between what will be the bundling approach and the solution for describing individual RTP streams attributes.

This document discusses the different RTP topologies and describes existing tools and see what they provide and how they can be extended to provide better SDP support for SSRC multiplexed RTP streams while supporting the different topologies.

## 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. RTP topologies for CLUE

The typical RTP topologies used by Telepresence systems specify different behaviors for RTP and RTCP distribution. A number of RTP topologies are described in [I-D.westerlund-avtcore-rtp-topologies-update]. The CLUE WG direction is to be able to support the relevant topologies including point-to-point, as well as media mixers, media-switching mixers, and source-projection mixers.

In the point-to-point topology, one peer communicates directly with a single peer over unicast. There can be one or more RTP sessions, and each RTP session can carry multiple RTP streams identified by their SSRC. All SSRCs will be recognized by the peers based on the information in the RTCP SDES report that will include the CNAME and SSRC of the sent RTP streams. In some cases, a video conferencing system with multiple video sources in a point-to-point may nonetheless have RTP which is best described by one of the mixer topologies below. For example, it can produce composed or switched RTP streams to be used by a receiving system with fewer displays than the sender has sources.

In the Media Mixer topology, the peers communicate only with the mixer. The mixer provides mixed or composed media streams, using its own SSRC for the sent streams. There are two cases here. In the first case the mixer may have separate RTP sessions with each peer (similar to the point to point topology) terminating the RTCP sessions on the mixer; this is known as Topo-RTCP-Terminating MCU in [I-D.westerlund-avtcore-rtp-topologies-update]. In the second case, the mixer can use a conference-wide RTP session similar to [I-D.westerlund-avtcore-rtp-topologies-update] Topo-mixer or Topo-Video-switching. The major difference is that for the second case, the mixer uses conference-wide RTP sessions, and distributes the RTCP reports to all the RTP session participants, enabling them to learn all the CNAMEs and SSRCs of the participants and know the contributing source or sources (CSRCs) of the original streams from the RTP header. In the first case, the Mixer terminates the RTCP and the participants cannot know all the available sources based on the RTCP information. The conference roster information including conference participants, endpoints, media and media-id (SSRC) can be available using the conference event package [RFC4575] element.

In the Media-Switching Mixer topology, the peer to mixer communication is unicast with mixer RTCP feedback. It is conceptually similar to a composing mixer as described in the previous paragraph, except that rather than composing or mixing multiple sources, the mixer provides one or more conceptual sources selecting one source at a time from the original sources. The Mixer creates a conference-wide RTP session by sharing remote SSRC values as CSRCs to all conference participants.

In the Source-Projection Mixer (SPM) topology, the peer to mixer communication is unicast with RTCP mixer feedback. Every potential sender in the conference has a source which is "projected" by the mixer into every other session in the conference; thus, every original source is maintained with an independent RTP identity to every receiver, maintaining separate decoding state and its original RTCP SDES information. However, RTCP is terminated at the mixer, which might also perform reliability, repair, rate adaptation, or transcoding on the stream. Senders' SSRCs may be renumbered by the mixer. The sender may turn the projected sources on and off at any time, depending on which sources it thinks are most relevant for the receiver; this is the primary reason why this topology must act as an RTP mixer rather than as a translator, as otherwise these disabled sources would appear to have enormous packet loss. Source switching is accomplished through this process of enabling and disabling projected sources, with the higher-level semantic assignment of reason for the RTP streams assigned externally.

When looking at SSRC multiplexing we can see that in various



topologies, the SSRC behavior may be different:

1. The SSRCs are static (assigned by the MCU/Mixer), and there is an SSRC for each media capture encoding defined in the CLUE protocol. Source information may be conveyed using CSRC, or, in the case of topo-RTCP-Terminating MCU, is not conveyed.
2. The SSRCs are dynamic, representing the original source and are relayed by the Mixer/MCU to the participants.

In the source projecting mixer (SPM) topology, the number of sources and their SSRCs may change dynamically. An example is a video conference that starts with 4 participants and the (SPM) forwards the video RTP streams from 3 of them to all participants. Later 10 more participants join the conference and the SPM will forward 9 video sources to each participant. The projected streams keep their original SSRCs and each participant may get different streams relayed by the SPM. The SPM creates a separate RTP session with each participant and will convey the origin of the media using RTCP SDES information. In this case the number of RTP streams and the sources they are coming from may change dynamically. This will be a challenge if we will need to explicitly provide in the SDP all the sources in the initial offer, and change it whenever a party joins or leaves. There is also a scaling issue to explicitly list all the sources for large conferences.

#### 4. Review of current directions in MMUSIC, AVText and AVTcore

This section provides an overview of the RFCs and drafts that tries to provide more information about RTP streams based on their SSRC and can be helpful to assign attribute to individual RTP streams that are multiplexed to a single transport address.

When looking at the available tools based on current work in MMUSIC, AVTcore and AVText for supporting SSRC multiplexing at the SDP level the following documents are considered to be relevant.

SDP Source attribute [RFC5576] mechanisms to describe specific attributes of RTP sources based on their SSRC. This document defines a mechanism to describe RTP sources, identified by their synchronization source(SSRC) identifier, in SDP, to associate attributes with these sources, and to express relationships among individual sources. It also defines a number of new SDP attributes that apply to individual sources ("source-level" attributes), describes how a number of existing media stream ("media-level") attributes can also be applied at the source level, and establishes IANA registries for source-level attributes and source grouping

semantics. This mechanism provides an extensible framework but that implies that there will be a need to specify source level attributes and probably to change the IANA procedure for attribute registration adding the requirement to specify if it is also source level attribute ( currently [RFC4566] requires for type of attribute to specify (session level, media level, or both))

[I-D.westerlund-mmusic-max-ssrc] a signaling solution for how to use multiple SSRCS within one RTP session. This document also defines two new SDP attributes, "max-send-ssrc" and " max-recv-ssrc". The attributes allows an entity to, for a given media description, indicate sending and receiving capabilities of multiple media sources, based on codec usage . Since the number of payload type numbers in an SDP m-line specifies that all these payloads can be received this draft provides a way to specify how many can be sent and received simultaneously. Still if there are more payload type numbers in the m-line it is still implies that a receiver must be able to receive any subset at any given time but no more than max-ssrc.

A proposed solution to support simulcast is defined in [I-D.westerlund-avtcore-rtp-simulcast]. Simulcast is an application usage where multiple media streams derived from the same media source may be sent simultaneously. The document discusses the best way of accomplishing this in RTP using a session-based solution. The document describes a solution where each stream from the unicast stream will use a separate RTP session. Section 4.2 of the document looks at using a single RTP session using RFC5576 [RFC5576] and the proposed source name attribute specified in [I-D.westerlund-avtext-rtcp-sdes-srcname]. Another way for a single session support may be by using a different payload type numbers but section 4.1 of [I-D.westerlund-avtcore-rtp-simulcast] discourages such usage.

[I-D.westerlund-avtext-rtcp-sdes-srcname] provides an extension that may be send in SDP, as an RTCP SDES information or as an RTP header extension that uniquely identifies a single media source. It defines a hierarchical order of the SRCNAME parameter that can be used, for example, to describe multiple resolutions from the same source (see section 5.1 of [I-D.westerlund-avtcore-rtp-simulcast]). Still all the examples are using RTP session multiplexing and there is no description of using a single RTP session. This can probably be addressed using bundle with separate m-line for each resolution.

Other documents that discusses the media source issue and may be required as part of the solution includes:

[I-D.lennox-mmusic-sdp-source-selection] specifies how participants

in a multimedia session can request a specific source from a remote party.

[I-D.westerlund-avtext-codec-operation-point] extends the codec control messages by specifying messages that let participants communicate a set of codec configuration parameters.

Negotiation of generic image attributes in SDP [RFC6236] provides the means to negotiate the image size. The image attribute can be used to offer different image parameters like size but in order to offer multiple RTP streams with different resolutions it does it using separate RTP session for each image option.

## 5. Requirements from a solution

When two or more endpoints with multiple sources communicate with each other and requires multiplexing multiple media types in the same RTP session, the following requirements should be addressed:

- o It should be possible to group relationships among sources of an RTP session
- o It should be possible to identify streams among sources of an RTP session
- o It should be possible to indicate the maximum number of Sources and Receivers
- o It should be possible to negotiate Codec Configuration parameters
- o It should be possible to request the transmission of specific sources
- o It should be possible to indicate the priority of transmission of sources
- o It should be possible to indicate support of multiple media type multiplexing
- o It should be possible to support receipt of multiple RTP sources without explicit per-source signaling or negotiation.
- o It must be possible for a multimedia session to use multiple transport flows for a given media type where it is considered valuable (for example, for distributed media, or differential quality-of-service).

- o It must be possible for a source to be placed into a switched RTP session even if the source is a "late joiner", i.e. was added to the conference after the receiver requested the switched source.
- o It must be possible for a receiver to identify the actual source that is currently being mapped to a switched media stream, and correlate it with out-of-band information such as rosters.
- o If a given source is being sent on the same transport flow (media track) for more than one reason (e.g. if it corresponds to more than one RTCWen Mediastream at once), it should be possible for a sender to send only one copy of the source.
- o On the network, media flows should, as much as possible, look and behave like currently-defined usages of existing protocols; established semantics of existing protocols must not be redefined.
- o The solution should seek to minimize the processing burden for boxes that distribute media to decoding hardware.
- o If multiple sources from a single synchronization context are being sent simultaneously, it must be possible for a receiver to associate and synchronize them properly.

## 6. SDP limitations and proposed solution

As the default behavior, Group relationship among sources of an RTP session can be indicated by extending the Session Description Protocol (SDP) Grouping Framework [RFC5888]. However the Session Description Protocol (SDP) Grouping Framework is limited to one media description per SFP m-line and does not support multiplexing multiple media types in one RTP session. Alternatively, Group relationship among sources of an RTP session can be implicitly indicated using hierarchical order of the SRCNAME parameter defined in [I-D.westerlund-avtext-rtcp-sdes-srcname].

Normally each RTP stream in the multiplexed RTP streams is identified by its SSRC. However in some cases, one media stream may include multiple sub-stream sharing the same properties, e.g., scalable media. In such cases, the capability to identify each sub-stream among sources of an RTP session is required.

[I-D.westerlund-avtext-codec-operation-point] provides a means for sub-stream identification. However such means are too much codec specific and used together with codec control messages.

It is clear that the current SDP does not provide enough tools to address all requirements. There are a couple of options to represent

multiple media streams, The major two options are:

- o Use multiple m-line each describing a single RTP stream
- o Use multiple m-lines each describing one or multiple RTP streams.

#### 6.1. single RTP stream

When using a single RTP stream in each m-line provides an easy way to describe the streams. This solution requires that all RTP media streams **MUST** be declared explicitly in the initial offer or in a later one before they can be used. As discussed above this solution does not scale well when doing a multipoint conference using the Source Projecting Mixer when the conference include multiple participants and each participant may get a different subset of all streams. Supporting this use case may require a lot of SIP re-invites.

#### 6.2. One or multiple RTP streams

For this option each m-line will specify the maximum number of SSRC that can be sent or received using this m-line. So similar streams can be added or removed implicitly without requiring more signaling. The solution will use the maxssrc attribute to specify how many RTP streams can be sent or received. If there is no maxssrc parameter it will imply a single RTP stream is specified. This option allows the SDP to use a single RTP stream per m-line if there is a need to specify specific attribute that cannot be described in a single m-line and bundle the m-lines.

### 7. Acknowledgements

Place Holder

### 8. IANA Considerations

TBD

### 9. Security Considerations

TBD.

### 10. References

## 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

## 10.2. Informative References

- [I-D.ietf-avtcore-multi-media-rtp-session]  
Westerlund, M., Perkins, C., and J. Lennox, "Multiple Media Types in an RTP Session",  
draft-ietf-avtcore-multi-media-rtp-session-01 (work in progress), October 2012.
- [I-D.lennox-mmusic-sdp-source-selection]  
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-04 (work in progress), March 2012.
- [I-D.westerlund-avtcore-rtp-simulcast]  
Westerlund, M., Burman, B., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP sessions",  
draft-westerlund-avtcore-rtp-simulcast-01 (work in progress), July 2012.
- [I-D.westerlund-avtcore-rtp-topologies-update]  
Westerlund, M. and S. Wenger, "RTP Topologies",  
draft-westerlund-avtcore-rtp-topologies-update-01 (work in progress), October 2012.
- [I-D.westerlund-avtext-codec-operation-point]  
Westerlund, M., Burman, B., and L. Hamm, "Codec Operation Point RTCP Extension",  
draft-westerlund-avtext-codec-operation-point-00 (work in progress), March 2012.
- [I-D.westerlund-avtext-rtcp-sdes-srcname]  
Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDES Item SRCNAME to Label Individual Sources",  
draft-westerlund-avtext-rtcp-sdes-srcname-01 (work in progress), July 2012.
- [I-D.westerlund-mmusic-max-ssrc]  
Holmberg, C., Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization Sources (SSRC) in SDP Media Descriptions", draft-westerlund-mmusic-max-ssrc-00 (work in progress), September 2012.

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.
- [RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 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.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, 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.

Authors' Addresses

Roni Even  
Huawei Technologies  
Tel Aviv,  
Israel

Email: [roni.even@mail01.huawei.com](mailto:roni.even@mail01.huawei.com)

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

Email: [jonathan@vidyo.com](mailto:jonathan@vidyo.com)

Qin Wu  
Huawei Technologies

Email: [bill.wu@huawei.com](mailto:bill.wu@huawei.com)





MMUSIC Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 22, 2013

C. Holmberg  
Ericsson  
H. Alvestrand  
Google  
C. Jennings  
Cisco  
February 18, 2013

Multiplexing Negotiation Using Session Description Protocol (SDP) Port  
Numbers  
draft-ietf-mmusic-sdp-bundle-negotiation-03.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 to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 2013.

Copyright Notice

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

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

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Conventions . . . . .	4
4. Applicability Statement . . . . .	4
5. SDP Grouping Framework BUNDLE Extension Semantics . . . . .	4
6. SDP Offer/Answer Procedures . . . . .	4
6.1. General . . . . .	5
6.2. SDP Offerer Procedures . . . . .	5
6.3. SDP Answerer Procedures . . . . .	7
6.4. Bundled SDP Information . . . . .	7
6.4.1. General . . . . .	7
6.4.2. Bandwidth (b=) . . . . .	7
6.4.3. Attributes (a=) . . . . .	7
7. Single vs Multiple RTP Sessions . . . . .	8
7.1. General . . . . .	8
7.2. Single RTP Session . . . . .	8
8. Usage With ICE . . . . .	8
8.1. General . . . . .	8
8.2. Candidates . . . . .	9
8.3. Candidates . . . . .	9
9. Security Considerations . . . . .	9
10. Example: SDP Offer with different port number values . . . . .	9
11. Example: SDP Offer with identical port number values . . . . .	11
12. IANA Considerations . . . . .	13
13. Acknowledgements . . . . .	13
14. Change Log . . . . .	13
15. References . . . . .	14
15.1. Normative References . . . . .	14
15.2. Informative References . . . . .	14
Appendix A. Design Considerations . . . . .	15
A.1. General . . . . .	15
A.2. UA Interoperability . . . . .	15
A.3. Usage of port number value zero . . . . .	16
A.4. B2BUA And Proxy Interoperability . . . . .	17
A.4.1. Traffic Policing . . . . .	18
A.4.2. Bandwidth Allocation . . . . .	18
A.5. Candidate Gathering . . . . .	18
Authors' Addresses . . . . .	18

## 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 SDP Offerer and SDP Answerer [RFC3264] use the "BUNDLE" grouping extension to indicate which media is associated with a single 5-tuple. For each media, the associated "m=" line is associated with a "BUNDLE" group.

For each "BUNDLE" group, the SDP Offerer and SDP Answerer use an identical port number value in each "m=" line associated with the "BUNDLE" group. However, until it is known that the SDP Answerer supports the "BUNDLE" grouping extension, when the SDP Offerer generates an SDP Offer, a different port number value is inserted for each "m=" line associated with the "BUNDLE" group. The port number value associated with the first "m=" line associated with the "BUNDLE" group represents the port value number offered to be used in the single 5-tuple.

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 different port number value MUST be used.

When media is transported using the Real-Time Protocol (RTP) [RFC3550], the default assumption of the mechanism is that all media associated with a "BUNDLE" group will form a single RTP Session [RFC3550]. However, future specifications can extend the mechanism, in order to negotiate RTP Session multiplexing, i.e. "BUNDLE" groups where media associated with a group form multiple RTP Sessions.

The mechanism is backward compatible. Endpoints that do not support the "BUNDLE" grouping extension are expected to generate SDP Offers and SDP Answers without inserting a "BUNDLE" group, and to insert different port number values in each "m=" line, in the SDP Offers and SDP Answers, as defined in RFC 4566 and RFC 3264.

## 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.

## 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 an SDP media-level 'mid' attribute, is part of a specific "BUNDLE" group.

## 6. SDP Offer/Answer Procedures

## 6.1. General

When an endpoint generates an SDP Offer or SDP Answer, which contains a "BUNDLE" group, it MUST insert an SDP session-level 'group' attribute, with a "BUNDLE" value, and assign SDP media-level 'mid' attribute values for each "m=" line associated the "BUNDLE" group.

Until it is known whether the SDP Answerer supports the "BUNDLE" grouping extension, the SDP Offerer MUST, for each "m=" line associated with a "BUNDLE" group:

- o 1. Insert different port number values.
- o 2. Insert identical connection data ("c=" line) value.
- o 3. Insert different SDP 'rtcp' attribute value, when used.
- o 4. Insert different ICE candidate values, when used.
- o 5. Insert an SDP 'rtcp-mux' attribute.
- o 6. Insert identical DTLS-SRTP fingerprint attribute, when used
- o 7. Insert different SDES crypto attribute, when used

Once it is known that both endpoints support the "BUNDLE" grouping extension, the SDP Offerer and SDP Answerer MUST, for each "m=" line associated with a "BUNDLE" group:

- o 1. Insert identical port number values.
- o 2. Insert identical connection data ("c=" line) value.
- o 3. Insert identical SDP 'rtcp' attribute value, when used.
- o 4. Insert identical ICE candidate values, when used.
- o 5. Insert an SDP 'rtcp-mux' attribute.
- o 6. Insert identical DTLS-SRTP fingerprint attribute, when used
- o 7. Insert different SDES crypto attribute, when used

Once both endpoints have indicated support of the "BUNDLE" grouping extension, they can start using the single port for the media associated with each "m=" line in the "BUNDLE group".

OPEN ISSUE #1: If the SDP Answerer supports "BUNDLE", do we mandate that "m=" lines must be grouped in the same way in the SDP Answer as they are grouped in the SDP Offer?

## 6.2. SDP Offerer Procedures

When the SDP Offerer generates an SDP Offer that contains a "BUNDLE" group, the SDP Offer MUST be generated according to the procedures in Section 6.1.

If the associated SDP Answer contains a "BUNDLE" group, and the SDP Offer contained different port number values in each "m=" line associated with the "BUNDLE" group, the SDP Offerer MUST generate a

new SDP Offer, and insert an identical port number value for each "m=" line associated with the "BUNDLE" group. Unless ICE is used, the SDP Offerer MUST generate the new SDP Offer immediately when it has received the SDP Answer indicating that the SDP Answerer supports the "BUNDLE" grouping extension. If ICE is used, the SDP Offer can wait until the SDP Offer indicating that ICE is finished is sent, as defined in RFC 5245.

NOTE: The reason for sending the new SDP Offer, which includes an identical port number value in each "m=" line associated with the "BUNDLE" group, is to ensure that intermediary entities that look at SDP information e.g. for different type of policing functions, have the correct information regarding which ports will be used for media.

If the SDP Offer contains different port number values for each "m=" lines associated with the "BUNDLE" group, and if the associated SDP Answer does not contain a "BUNDLE" group, the SDP Offerer MUST use the different port number values that were included in the SDP Offer.

Once it is known that both the SDP Offerer and SDP Answerer support the "BUNDLE" grouping extension, in each subsequent SDP Offer towards the SDP Answerer, the SDP Offer MUST insert an identical port number value in each "m=" line associated with the "BUNDLE" group.

NOTE: If needed, the SDP Offerer is allowed to change the port number value in an subsequent SDP Offer, but it still inserts the same identical port number value in each "m=" line associated with the "BUNDLE" group.

When generating an SDP Offer, if the SDP Offerer wants to disable media associated with an "m=" line in a "BUNDLE" group, it will insert a zero port number value in the disabled "m=" line, as defined in RFC 3264. However, if the "m=" lines associated with the "BUNDLE" group contain different port number values (i.e. the SDP Offer is generated until it is known whether the SDP Answerer supports the "BUNDLE" grouping extension) the SDP Offerer MUST NOT set the port number value of the first "m=" line associated with the "BUNDLE" group to zero.

The SDP Offerer MUST NOT insert an identical port number value in multiple "m=" lines, unless the "m=" lines are associated with a "BUNDLE" group.

OPEN ISSUE #2: If the session has been established without BUNDLE, do we allow BUNDLE to be enabled later during the session?

OPEN ISSUE #3: If the session has been established with BUNDLE, do we allow BUNDLE to be disabled later during the session, meaning that

different port number values will be used for media associated with each "m=" line?

### 6.3. SDP Answerer Procedures

When an SDP Answerer receives an SDP Offer which contains a "BUNDLE" group, and the SDP Answerer accepts the offered "BUNDLE" group, the SDP Answerer MUST generate an SDP Answer according to the procedures in Section 6.1.

When generating the SDP Answer, if the SDP Answerer wants to reject media associated with an "m=" line in the "BUNDLE" group, it will insert a zero port number value in the disabled "m=" line, as defined in RFC 3264.

The SDP Answerer MUST NOT insert a "BUNDLE" group in an SDP Answer, unless the associated SDP Offer contains a "BUNDLE" group.

The SDP Answerer MUST NOT insert an identical port number value in multiple "m=" lines, unless the "m=" lines are associated with a "BUNDLE" group.

Until the SDP Answerer receives the new SDP Offer, which contains an identical port number value for each "m=" line associated with a "BUNDLE" group, it MUST use the port, and related information (ICE candidates, SDES keys etc) associated with the first "m=" line in the "BUNDLE" group.

### 6.4. Bundled SDP Information

#### 6.4.1. General

This section describes how SDP information, given for each media description, is calculated into a single value for a "BUNDLE" group.

#### 6.4.2. Bandwidth (b=)

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

#### 6.4.3. Attributes (a=)

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



## 7. Single vs Multiple RTP Sessions

### 7.1. General

When entities negotiate the usage of bundled media, the default assumption is that all media associated with the bundled media will form a single RTP session.

The usage of multiple RTP Sessions within a "BUNDLE" group is outside the scope of this specification. Other specification needs to extend the mechanism in order to allow negotiation of such bundle groups.

It is possible to use multiple "BUNDLE" groups, in case the RTP media within each group will be part of different RTP Sessions.

### 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 SDP 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 SDP 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. Example: SDP Offer with different port number values

The example below shows an SDP Offer, where bundled media is offered using different port number values in the "m=" lines associated with the "BUNDLE" group. The example also shows two SDP Answer alternatives: one where bundled media is accepted, and one where bundled media is rejected (or, not even supported) by the SDP Answerer.

SDP Offer (Bundled media offered)

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

SDP Answer (Bundled media accepted)

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

SDP Answer (Bundled media not accepted)

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

SDP Offer with ICE (Bundled media offered)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=candidate:1 1 UDP 1694498815 host.atlanta.com 10000 typ host
m=video 10002 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=candidate:1 1 UDP 1694498815 host.atlanta.com 10002 typ host
```

#### 11. Example: SDP Offer with identical port number values

The example below shows an SDP Offer, where bundled media is offered an identical port number value in the "m=" lines associated with the "BUNDLE" group. The example also shows two SDP Answer alternatives: one where bundled media is accepted, and one where bundled media is rejected (or, not even supported) by the SDP Answerer.

SDP Offer (Bundled media offered)

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

SDP Answer (Bundled media accepted)

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

SDP Answer (Bundled media not accepted)

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.com
s=
c=IN IP4 host.biloxi.com
t=0 0
m=audio 20000 RTP/AVP 0
b=AS:200
```

```
a=rtpmap:0 PCMU/8000
m=video 30000 RTP/AVP 32
b=AS:1000
a=rtpmap:32 MPV/90000
```

SDP Offer with ICE (Bundled media offered)

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.com
s=
c=IN IP4 host.atlanta.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=candidate:1 1 UDP 1694498815 host.atlanta.com 10000 typ host
m=video 10000 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=candidate:1 1 UDP 1694498815 host.atlanta.com 10000 typ host
```

## 12. IANA Considerations

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

## 13. Acknowledgements

The usage of the SDP grouping extension for negotiating bundled media is based on a similar alternative proposed by Harald Alvestrand. The SDP examples are also modified versions from the ones in the Alvestrand proposal.

## 14. Change Log

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

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

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

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

- o No changes. New version due to expiration.

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

- o No changes. New version due to expiration.

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

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

## 15. References

### 15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [I-D.nandakumar-mmusic-sdp-attributes]  
Nandakumar, S. and C. Jennings, "A Framework for SDP Attributes when Multiplexing",  
draft-nandakumar-mmusic-sdp-attributes-00 (work in progress), February 2013.

### 15.2. 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.

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

## Appendix A. Design Considerations

### A.1. General

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

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

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

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

### A.2. UA Interoperability

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

SDP Offer

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



## SDP Answer

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.com
s=
c=IN IP4 host.biloxi.com
t=0 0
m=audio 20000 RTP/AVP 97
a=rtpmap:97 iLBC/8000
m=video 20002 RTP/AVP 97
a=rtpmap:97 H261/90000
```

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

## SDP Offer

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

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

## A.3. Usage of port number value zero

In an SDP Offer or SDP Answer, the media associated with an "m=" line can be disabled/rejected by setting the port number value to zero.

This is different from e.g. using the SDP direction attributes, where RTCP traffic will continue even if the SDP "inactive" attribute is indicated for the associated "m=" line.

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

#### A.4. B2BUA And Proxy Interoperability

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

SDP Offer

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

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

#### A.4.1. Traffic Policing

Sometimes intermediaries do not act as B2BUA, in the sense that they don't modify SDP bodies, nor do they terminate SIP dialogs. Still, however, they may use SDP information (e.g. IP address and port) in order to control traffic gating functions, and to set traffic policing rules. There might be rules which will trigger a session to be terminated in case media is not sent or received on the ports retrieved from the SDP. This typically occurs once the session is already established and ongoing.

#### A.4.2. Bandwidth Allocation

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

#### A.5. Candidate Gathering

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

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

Authors' Addresses

Christer Holmberg  
Ericsson  
Hirsalantie 11  
Jorvas 02420  
Finland

Email: christer.holmberg@ericsson.com

Harald Tveit Alvestrand  
Google  
Kungsbron 2  
Stockholm 11122  
Sweden

Email: harald@alvestrand.no

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

Email: fluffy@iii.ca



Network WG  
Internet-Draft  
Expires: August 18, 2013  
Intended Status: Standards Track (PS)

James Polk  
Subha Dhesikan  
Paul Jones  
Cisco Systems  
Feb 18, 2013

The Session Description Protocol (SDP) 'trafficclass' Attribute  
draft-ietf-mmusic-traffic-class-for-sdp-03

Abstract

This document proposes a new Session Description Protocol (SDP) attribute to identify the traffic class a session is requesting in its offer/answer exchange.

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 18, 2013.

Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Traffic Class Framework and Component Definitions . . . . .	5
3. Traffic Class Attribute Definition . . . . .	6
3.1 Categories within the SDP Traffic Class Label . . . . .	8
3.2 Applications within the SDP Traffic Class Label . . . . .	9
3.3 Adjectives within the SDP Traffic Class Label . . . . .	9
3.3.1 Qualified Adjectives . . . . .	9
4. Matching Categories with Applications and Adjectives . . . . .	11
4.1 Conversational Category Traffic Class . . . . .	11
4.2 Multimedia-Conferencing Category Traffic Class . . . . .	12
4.3 Realtime-Interactive Category Traffic Class . . . . .	14
4.4 Multimedia-Streaming Category Traffic Class . . . . .	15
4.5 Broadcast Category Traffic Class . . . . .	17
5. Offer/Answer Behavior . . . . .	18
5.1 Offer Behavior . . . . .	18
5.2 Answer Behavior . . . . .	19
6. Security considerations . . . . .	20
7. IANA considerations . . . . .	20
8. Acknowledgments . . . . .	23
9. References . . . . .	24
9.1. Normative References . . . . .	24
9.2. Informative References . . . . .	24
Authors' Addresses . . . . .	25
Appendix . . . . .	25

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].

## 1. Introduction

The Session Description Protocol (SDP) [RFC4566] provides a means for an offerer to describe the specifics of a session to an answerer, and for the answerer to respond back with its session specifics to the offerer. These session specifics include offering the codec or codecs to choose from, the specific IP address and port number the offerer wants to receive the RTP stream(s) on/at, the particulars about the codecs the offerer wants considered or mandated, and so on.

There are many facets within SDP to determine the Real-time Transport Protocol (RTP) [RFC3550] details for the session establishment between one or more endpoints, but identifying how the underlying network should process each stream still remains under-specified.

The ability to identify a traffic flow by port number gives an indication to underlying network elements to treat traffic with dissimilar ports in a different way, the same or in groups the same - but different from other ports or groups of ports.

Within the context of realtime communications, the labeling of an RTP session based on media descriptor lines as just a voice and/or video session is insufficient, and provides no guidelines to the underlying network on how to treat the traffic. A more granular labeling helps on several fronts to

- inform application layer elements in the signaling path the intent of this session.
- inform the network on how to treat the traffic if the network is configured to differentiate session treatments based on the type of session the RTP is, including the ability to provide call admission control based on the type of traffic in the network.
- allow network monitoring/management of traffic types realtime and after-the-fact analysis.

Some network operators want the ability to guarantee certain traffic gets a minimum amount of network bandwidth per link or through a series of links that make up a network such as a campus or WAN, or a backbone. For example, a call center voice application might get at least 20% of the available link bandwidth.

Some network operators want the ability to allow certain users or devices access to greater bandwidth during non-busy hours than during busy hours of the day. For example, all desktop video might operate at 1080p during non-peak times, but a similar session might be curtailed between the same users or devices to 720p or 360p during peak hours. Another example would be to reduce the frames per second (fps) rate, say from 30fps to 15fps. This case is not as clear as accepting or denying similar sessions during different times of the day, but tuning the access to the bandwidth based on the type of session. In other words, tune down the bandwidth for desktop video during peak hours to allow a 3-screen Telepresence session that would otherwise look like the same type of traffic (RTP, and more granular, video).

RFC 4594 established a guideline for classifying the various flows in the network and the Differentiated Services Codepoint (DSCP) values that apply to many traffic types (table 3 of [RFC4594]), including RTP based voice and video traffic sessions. The RFC also defined the per hop network behavior that is strongly encouraged for each of these application traffic types based on the traffic characteristics and tolerances to delay, loss and jitter within each traffic class.

Video was broken down into four categories in that RFC, and voice in



another single category. We do not believe this satisfies the technical and business requirements to accomplish sufficiently unique labeling of RTP traffic.

If the application becomes aware of traffic labeling,

- this can be coded into layer 3 mechanisms.
- this can be coded into layer 4 protocols and/or mechanisms.
- this can be coded into a combination of mechanisms and protocols.

The layer 3 mechanism for differentiating traffic is either the port number or the Differentiated Services Codepoint (DSCP) value [RFC2474]. Within the public Internet, if the application is not part of a managed service, the DSCP value likely will be best effort (BE), or reset to BE when ingressing a provider's network. Within the corporate LAN, this is usually completely configurable and a local IT department can take full advantage of this labeling to shape and manage their network as they see fit.

Within a network core, DiffServ typically does not apply. That said, DiffServ can be used to identify which traffic goes into which MPLS tunnel [RFC4124].

Labeling realtime traffic types using a layer 4 protocol would likely involve RSVP [RFC2205] or NSIS [RFC4080]. RSVP has an Application Identifier (app-ID) defined in [RFC2872] that provides a means for carrying a traffic class label along the media path. An advantage of this mechanism is that the label can inform each domain along the media path what type of traffic this traffic flow is, and allow each domain to adjust the appropriate DSCP value (set by each domain for use within that domain). Meaning, if a DSCP value is set by an endpoint or a router in the first domain and gets reset by a service provider, the far-end domain will be able to reset the DSCP value appropriate for the intended traffic class. There is a proposed extension to RSVP which creates individual profiles for what goes into each app-ID field to describe these traffic classes [ID-RSVP-PROF], which will take advantage of what is described in this document.

There are several proprietary mechanisms that can take advantage of this labeling, but none of those will be discussed here.

The idea of traffic - or service - identification is not new; it has been described in [RFC5897]. If that RFC is used as a guideline, identification that leads to stream differentiation can be quite useful. One of the points within RFC 5897 is that users cannot be allowed to assign any identification (fraud is one reason given). In addition, RFC 5897 recommends that service identification should be done in signaling, rather than guessing or deep packet inspection. Currently, any network would have to guess or perform deep

packet inspection to classify traffic and offer the service as per RFC 4594 as such service identification information is currently not available in SDP and therefore to the network elements. Since SDP understands how each stream is created (i.e., the particulars of the RTP stream), this is the right place to have this service differentiated. Such service differentiation can then be communicated to and leveraged by the network.

[Editor's Note: the words "traffic" and "service" are similar enough that the above paragraph talks about RFC 5897's "service identification", but this document only discuss and propose traffic indications in SDP.]

This document proposes a simple attribute line to identify the application a session is requesting in its offer/answer exchange. This document uses previously defined service class strings for consistency between IETF documents.

This document modifies the traffic classes originally created in RFC 4594 in Section 2, incrementing each class with application identifiers and optional adjective strings. Section 3 defines the new SDP attribute "trafficclass". Section 4 discusses the offerer and answerer behavior when generating or receiving this attribute.

## 2. Traffic Class Framework and Component Definitions

The framework of the traffic class attribute will have at least two parts, called components, allowing for several more to be included further distinguishing a particular session's traffic classification from another session's traffic classification. The amount of indicated differentiation between sessions is not a goal, and should only have additional components for differentiation if there is a need to uniquely identify traffic in different sessions.

The intention is to have a category component (e.g., conversational) that identifies the traffic pattern for a session. Is the traffic within a session one-way or two-way? Can the traffic be buffered before reaching the destination or not? What is this session's tolerance to packet loss and can there be retransmissions?

The application component (e.g., video) identifies the basic type of traffic within a category. Is it media or data packets? If media, which type of media? If data packets, which application of data packets are in this session?

The optional adjective component(s) (e.g., immersive) help to further refine the traffic within a session by providing more description. For instance, if a session is two-way voice, what additional information can be given about this particular session to refine its description? Is it part of a conference or telepresence session? Is it just standalone voice call? Has a capacity admission

protocol or mechanism been applied to this session?

The traffic class label will have the following structure,

```
category.application(.adjective)(.adjective)...
```

[Editor's Note: the above is not the exact ABNF to be used.  
The order is right. The category and application  
MUST appear first (each only once) and zero or more  
adjectives can appear following the application  
component.]

Where

- 1) the 1st component is the category, and is mandatory;
- 2) the 2nd component is the application, and is mandatory;
- 3) an optional 3rd component or series of components are  
adjective(s) used to further refine the application component;

The construction of the traffic class label for Telepresence video  
would follow the minimum form of:

```
conversational.video.immersive
```

where there might be one or more adjective after '.immersive'.

There is no traffic class or DSCP value associated with just  
"conversational". There is a traffic class associated with  
"conversational.video", creating a differentiation between it and a  
"conversational.video.immersive" traffic class, which would have  
DSCP associated with the latter traffic class, depending on local  
policy. Each category component is defined below, as are several of  
application and adjective strings.

### 3. Traffic Class Attribute Definition

This document proposes the 'trafficclass' session and media-level  
SDP attribute. The following is the Augmented Backus-Naur Form  
(ABNF) [RFC5234] syntax for this attribute, which is based on the  
SDP [RFC4566] grammar:

```
attribute                =/ traffic-class-label

traffic-class-label      = "trafficclass" ":" [SP] category
                          "." application *( "." adjective )

category                 = "broadcast" /
                          "realtime-interactive" /
                          "multimedia-conferencing" /
                          "multimedia-streaming" /
                          "conversational" / tcl-token
```

```

application          = tcl-token

adjective            = classified-adjective /
                      unclassified-adjective

classified-adjective  = tcl-token ":" tcl-token

unclassified-adjective = tcl-token

tcl-token            = ALPHA *( [ "-" ] ALPHA / DIGIT )

```

The attribute is named "trafficclass", for traffic classification, identifying which one of the five categories applies to the media stream associated with this m-line. There MUST NOT be more than one category component per media line.

The categories in this document are an augmented version of the application labels introduced by table 3 of RFC 4595 (which will be rewritten based on the updated labels and treatments expected for each traffic class defined in this document).

Application Labels Defined in RFC 4594	Category Classes Defined in this document
broadcast-video	broadcast
realtime-interactive	realtime-interactive
multimedia-conferencing	multimedia-conferencing
multimedia-streaming	multimedia-streaming
telephony	conversational

Figure 1. Label Differences from RFC 4594

As is evident from the changes above, from left to right, two labels are different and each of the meanings are different in this document relative to how RFC 4594 defined them. These differences are articulated in Section 4 of this document.

Applications and adjectives are defined using the syntax of "tcl-token" defined above.

RFC 4566 defined SDP as case sensitive. Everything is here as well.

An algorithm such as alphabetizing the list of components and matching the understood strings SHOULD be used for determining the traffic within a session.

Any category, application, or adjective string component within this attribute that is not understood MUST be ignored, leaving all that is understood to be processed. Ignored components SHOULD NOT be deleted, as a downstream entity could understand the component(s) and use it/them during processing.

The following is an example of media level description with a 'trafficclass' attribute:

```
m=video 50000 RTP/AVP 112
a=trafficclass conversational.video.immersive.aq:admitted
```

The above indicates the video part of a Telepresence session that has had capacity admission process applied to its media flow.

### 3.1 Categories within the SDP Traffic Class Label

The category component within the traffic class attribute describes the type of communication that will occur within that session. It answers these questions, is the traffic

- one-way or two-or-more-way interactive?
- elastic or inelastic (as far as retransmissions)?
- buffered or (virtually) non-buffered?
- media or non-media (data)?

The five category components of the traffic class attribute defined within this specification are as follows:

- o conversational
- o multimedia-conferencing
- o realtime-interactive
- o multimedia-streaming
- o broadcast

Sections 3.1 through 3.5 define each of the above.

The category component MUST NOT be the only component present in a traffic class attribute. The category component MUST BE paired with an application component to give enough meaning to the traffic class labeling goal.

Not understanding the category component SHOULD mean that this attribute is ignored, because of the information about the communication flow within that component.

### 3.2 Applications within the SDP Traffic Class Label

The application component identifies the application of a particular traffic flow, for example, audio or video. The application types are listed and defined in Section 2 of this document. Not every category is paired with every application listed, at least as defined in this document. One or more applications are inappropriate in one or more categories. For example, iptv is a single directional traffic application that is suited for the broadcast (one-way) category rather than categories like realtime-interactive or conversational.

Section 4.1 through 4.5 list many of the expected combinations.

### 3.3 Adjectives within the SDP Traffic Class Label

For additional application type granularity, adjective components can be added. One or more adjectives can be within the same traffic class attribute to provide more differentiation.

It is important to note that while the order of component types matter, the order of the adjective components do not. There might be local significance to the ordering of adjectives though, such as having a pattern matching algorithm in which labels are matched exactly (i.e., the order matters), or not at all. In other words, the category class component **MUST** be before the application component, which **MUST** be before any and all adjective component(s).

There is no limit to the number of adjectives allowed.

Adjective components come in two versions, unqualified and qualified. One has a prefix (qualified), the other (unqualified) does not. A defined qualified adjective **MUST NOT** appear without its qualifier name, even in future extensions to this specification. Some implementations will likely perform a search within this attribute for the presence of qualifiers, which might be as simple as searching for the ":" COLON character. Implementations will be confused with inconsistent coding, therefore strict adherence is necessary.

#### 3.3.1 Qualified Adjectives

Adjectives can be either unqualified or qualified. Qualified adjectives have a delimiter ":" character between the "qualifier name" and the "qualifier value". As one example, we introduce in this specification the "admission qualifier" and it has a qualifier name of "aq". We also define several possible qualifier values for the admission qualifier, namely "admitted", "non-admitted", "partial", and "none". When present in a TCL string, the qualified adjectives look like these admission qualifier adjectives:

aq:admitted  
aq:non-admitted  
aq:partial  
aq:none

Defining some adjectives as qualified adjectives allows entities processing the traffic class label to potentially recognize a particular qualifier name and act on it, even if it does not understand the qualifier value. In the future, a new admission qualifier value might be defined, e.g. "foo", and entities could at least recognize the admission qualifier adjective, even if it did not understand the qualifier value "foo".

Like all adjectives, it is OPTIONAL to include the admission qualifier adjective in any trafficclass attribute.

The admission qualifier and its qualifier values are defined as:

- aq - 'admission qualifier' - this is the qualifier name for the admission qualifier adjectives, wherein the following qualifier values indicate the admission status for the traffic flow described by this m-line.
- admitted - capacity admission mechanisms or protocols are to be or were used for the full amount of bandwidth in relation to this m= line.
- non-admitted - capacity admission mechanisms or protocols were attempted but failed in relation to this m= line. This does not mean the flow described by this m= line failed. It just failed to attain the capacity admission mechanism or protocol necessary for a predictable quality of service, and is likely to continue with only a class of service marking or best effort.
- partial - capacity admission mechanisms or protocols are to be or were used for the part of the amount of bandwidth in relation to this m= line. All traffic above a certain amount will have no capacity admission mechanisms applied. In other words, there is more traffic sent than was agreed to. The burden is on the sender and receiver to deal with any sent and lost information.
- none - no capacity admission mechanisms or protocols are or were attempted in relation to this m= line.

The default for any flow generated from an m-line not having a trafficclass adjective of 'aq:admitted' or 'aq:non-admitted' MUST be the equivalent of 'aq:none', whether or not it is present.

#### 4. Matching Categories with Applications and Adjectives

This section describes each component within this document, as well as provides the combinations of categories and applications and adjectives. Given that not every combination makes sense, we express the limits here - which will be IANA registered.

##### 4.1 Conversational Category Traffic Class

The "conversational" traffic class is best suited for applications that require very low delay variation and generally intended to enable realtime, bi-directional person-to-person or multi-directional via an MCU communication. Conversational flows are inelastic, and with few exceptions, use a UDP transport.

Traffic Class Name	Traffic Characteristics	Tolerance to		
		Loss	Delay	Jitter
conversational	High priority, typically small packets (large video frames produce large packets), generally sustained high packet rate, low inter-packet transmission interval, usually UDP framed in (S)RTP	Very Low	Very Low	Very Low

Figure 2. Conversational Traffic Characteristics

The following application components are appropriate for use with the Conversational category:

- o audio (voice)
- o video
- o text (i.e., real-time text required by deaf users)
- o multiplex (i.e., combined a/v streams)

With adjective substrings to the above

immersive (TP) - An interactive audio-visual communications experience between remote locations, where the users enjoy a strong sense of realism and presence between all participants by optimizing a variety of attributes such as audio and video quality, eye contact, body language, spatial audio, coordinated environments and natural image size.



avconf - An interactive audio-visual communication experience that is not immersive in nature, though can have a high resolution video component.

text - a term for real-time transmission of text in a character-by-character fashion for use in conversational services, often as a text equivalent to voice-based conversational services. Conversational text is defined in the ITU-T Framework for multimedia services, Recommendation F.700 [RFC5194].

Multiplex - an application wherein media of different forms (e.g., audio and video) is multiplexed within the same media flow.

Category	Application	Adjective
conversational	audio	immersive
		avconf
		aq:admitted
		aq:non-admitted
		aq:partial
		aq:none
	video	immersive
		avconf
		aq:admitted
		aq:non-admitted
		aq:partial
		aq:none
	text	aq:admitted
		aq:non-admitted
		aq:partial
		aq:none
	multiplex	aq:admitted
		aq:non-admitted
		aq:partial
		aq:none

Figure 3. Conversational Applications and Adjective Combinations

#### 4.2 Multimedia-Conferencing Category Traffic Class

The "multimedia-conferencing" traffic class is best suited for applications that are generally intended for communication between human users, but are less demanding in terms of delay, packet loss,

and jitter than what conversational applications require. These applications require low to medium delay and may have the ability to change encoding rate (rate adaptive) or transmit data at varying rates.

Traffic Class Name	Traffic Characteristics	Tolerance to		
		Loss	Delay	Jitter
multimedia- conferencing	Variable size packets, Variable transmit interval, rate adaptive, reacts to loss, usually TCP-based	Low	Low	Low
		-	-	-
		Medium	Medium	Medium

Figure 4. Multimedia Conferencing Traffic Characteristics

Multimedia-conferencing flows are not to be media based. Media sessions use other categories. Multimedia-conferencing flows are those data flows that are typically transmitted in parallel to currently active media flows. For example, a two-way conference session in which the users share a presentation. The presentation part of that conference call uses the Multimedia-conferencing category, whereas the audio and any video uses the conversational category indication.

The following application components are appropriate for use with the Multimedia-Conferencing category:

- o application-sharing (that webex does or protocols like T.128) -  
An application that shares the output of one or more running applications or the desktop on a host. This can utilize vector graphics, raster graphics or video.
- o presentation-data - can be a series of still images or motion video.
- o whiteboarding - an application enabling the exchange of graphical information including images, pointers and filled and unfilled parametric drawing elements (points, lines, polygons and ellipses).
- o (RTP-based) file-transfer
- o instant messaging

Category	Application	Adjective
multimedia- conferencing	application-sharing	aq:admitted
		aq:non-admitted
		aq:partial

		aq:none
	whiteboarding	aq:admitted aq:non-admitted aq:partial aq:none
	presentation-data	aq:admitted aq:non-admitted aq:partial aq:none
	instant-messaging	aq:admitted aq:non-admitted aq:partial aq:none
	file-transfer	aq:admitted aq:non-admitted aq:partial aq:none

Figure 5. Multimedia Conferencing Applications and Adjective Combinations

#### 4.3 Realtime-Interactive Category Traffic Class

The "Realtime-Interactive" traffic class is intended for interactive variable rate inelastic applications that require low jitter and loss and very low delay. Many of the applications that use the Realtime-Interactive category use TCP or SCTP, even gaming, because lost packets is information that is still required - therefore it is retransmitted.

Traffic Class Name	Traffic Characteristics	Tolerance to		
		Loss	Delay	Jitter
realtime- interactive	Inelastic, mostly variable rate, rate increases with user activity	Low	Very Low	Low

Figure 6. Realtime Interactive Traffic Characteristics

The following application components are appropriate for use with the Realtime-Interactive category:

- o gaming - interactive player video games with other users on other

hosts (e.g., Doom)

- o remote-desktop - controlling a remote node with local peripherals (i.e., monitor, keyboard and mouse)
- o telemetry - a communication that allows remote measurement and reporting of information (e.g., post launch missile status or energy monitoring)

With adjective substrings to the above

- o virtual - To be used with the remote-desktop application component specifically when the traffic is a virtual desktop similar to an X-windows station, has no local hard drive, or is operating an computer application with no local storage.

Category	Application	Adjective
realtime-interactive	gaming	aq:admitted
		aq:non-admitted
		aq:partial
		aq:none
	remote-desktop	virtual
		aq:admitted
		aq:non-admitted
		aq:partial
		aq:none
	telemetry	aq:admitted
		aq:non-admitted
		aq:partial
		aq:none

Figure 7. Realtime-Interactive Applications and Adjective Combinations

#### 4.4 Multimedia-Streaming Category Traffic Class

The "multimedia-streaming" traffic class is best suited for variable rate elastic streaming media applications where a human is waiting for output and where the application has the capability to react to packet loss by reducing its transmission rate.

Traffic Class Name	Traffic Characteristics	Tolerance to		
		Loss	Delay	Jitter
multimedia-	Variable size packets,	Low -	Medium	High

streaming	elastic with variable rate	Medium	- High	
+-----+-----+-----+-----+				

Figure 8. Multimedia Streaming Traffic Characteristics

The following application components are appropriate for use with the Multimedia-Streaming category:

- o audio (see Section 4.1)
- o video (see Section 4.1)
- o webcast
- o multiplex (see Section 4.1)

The primary difference from the multimedia-streaming category and the broadcast category is about the length of time for buffering. Buffered streaming audio and/or video which are initiated by SDP, and not HTTP. Buffering here can be from many seconds to hours, and is typically at the destination end (as opposed to Broadcast buffering which is minimal at the destination). The buffering aspect is what differentiates this category class from the broadcast category (which has minimal or no buffering).

Category	Application	Adjective
multimedia-streaming	audio	aq:admitted aq:non-admitted aq:partial aq:none
	video	aq:admitted aq:non-admitted aq:partial aq:none
	webcast	live aq:admitted aq:non-admitted aq:partial aq:none
	multiplex	aq:admitted aq:non-admitted aq:partial aq:none

Figure 9. Multimedia Streaming Applications and Adjective Combinations

#### 4.5 Broadcast Category Traffic Class

The "broadcast" traffic class is best suited for inelastic streaming media Applications, which might have a 'wardrobe malfunction' delay at or near the source but not typically at the destination, that may be of constant or variable rate, requiring low jitter and very low packet loss.

See Section 4.4 for the difference between Multimedia-Streaming and Broadcast; it all has to do with buffering.

Traffic Class Name	Traffic Characteristics	Tolerance to		
		Loss	Delay	Jitter
broadcast	Constant and variable rate, inelastic, generally non-bursty flows, generally sustained high packet rate, low inter-packet transmission interval, usually UDP framed in (S)RTP	Very Low	Low - Medium	Low - Medium

Figure 10. Broadcast Traffic Characteristics

The following application components are appropriate for use with the Broadcast category:

- o audio (see Section 4.1)
- o video (see Section 4.1)
- o iptv
- o multiplex (see Section 4.1)

With adjective substrings to the above:

- o live (non-buffered)
- o surveillance - one way audio from a microphone or video from a camera (e.g., observing a parking lot or building exit), typically enabled for long periods of time, usually stored at the destination.

Category	Application	Adjective
broadcast	audio	surveillance live aq:admitted aq:non-admitted aq:partial aq:none
	video	surveillance live aq:admitted aq:non-admitted aq:partial aq:none
	iptv	live aq:admitted aq:non-admitted aq:partial aq:none
	multiplex	aq:admitted aq:non-admitted aq:partial aq:none

Figure 11. Broadcast Applications and Adjective Combinations

## 5. Offer/Answer Behavior

Through the inclusion of the 'trafficclass' attribute, an offer/answer exchange identifies the application type for use by endpoints within a session. Policy elements can use this attribute to determine the acceptability and/or treatment of that session through lower layers. One specific use-case is for setting of the DSCP specific for that application type (say a broadcast instead of a conversational video), decided on a per domain basis - instead of exclusively by the offering domain.

### 5.1 Offer Behavior

Offerers include the 'trafficclass' attribute with a single string comprised of two or more components (from the list in Section 2) to obtain configurable and predictable classification between the answerer and the offerer. The offerer can also include a private set of components, or a combination of IANA registered and private components within a single domain (e.g., enterprise networks).

Offerers of this 'trafficclass' attribute MUST NOT change the label in transit (e.g., wrt to B2BUAs). Session Border Controllers (SBC) at domain boundaries can change this attribute through local policy.

Offers containing a 'trafficclass' label not understood are ignored by default (i.e., as if there was no 'trafficclass' attribute in the offer).

## 5.2 Answer Behavior

Upon receiving an offer containing a 'trafficclass' attribute, if the offer is accepted, the answerer will use this attribute to classify the session or media (level) traffic accordingly towards the offerer. This answer does not need to match the traffic class in the offer, though this will likely be the case most of the time.

In order to understand the traffic class attribute, the answerer MUST check several components within the attribute, such as

- 1 - does the answerer understand the category component?

If not, the attribute SHOULD be ignored.

If yes, it checks the application component.

- 2 - does the answerer understand the application component?

If not, the answerer needs to check if it has a local policy to proceed without an application component. The default for this situation is as if the category component was not understood, the attribute SHOULD be ignored.

If yes, it checks to see if there are any adjective components present in this attribute to start its classification.

- 3 - does the answerer understand the adjective component or components if any are present?

If not present, process and match the trafficclass label value as is.

If yes, determine if there is more than one. Search for each that is understood. Any adjectives not understood are to be ignored, as if they are not present. Match all remaining understood components according to local policy and process attribute.

The answerer will answer the offer with its own 'trafficclass' attribute, which will likely be the same value, although this is not mandatory (at this time). The Offerer will process the received



answer just as the answerer processed the offer. In other words, the processing steps and rules are identical for each end.

The answerer should expect to receive RTP packets marked as indicated by its 'trafficclass' attribute in the answer itself.

An Answer MAY have a 'trafficclass' attribute when one was not in the offer. This will at least aid the local domain, and perhaps each domain the session transits, to categorize the application type of this RTP session.

Answerers that are middleboxes can use the 'trafficclass' attribute to classify the RTP traffic within this session however local policy determines. In other words, this attribute can help in deciding which DSCP an RTP stream is assigned within a domain, if the answerer were an inbound SBC to a domain.

## 6. Security considerations

RFC 5897 [RFC5897] discusses many of the pitfalls of service classification, which is similar enough to this idea of traffic classification to apply here as well. That document highly recommends the user not being able to set any classification. Barring a hack within an endpoint (i.e., to intentionally misclassifying (i.e., lying) about which classification an RTP stream is), this document's solution makes the classification part of the signaling between endpoints, which is recommended by RFC 5897.

## 7. IANA considerations

### 7.1 Registration of the SDP 'trafficclass' Attribute

This document requests IANA to register the following SDP att-field under the Session Description Protocol (SDP) Parameters registry:

Contact name: jmpolk@cisco.com

Attribute name: trafficclass

Long-form attribute name: Traffic Classification

Type of attribute: Session and Media levels

Subject to charset: No

Purpose of attribute: To indicate the Traffic Classification application for this session

Allowed attribute values: IANA Registered Tokens

Registration Procedures: Specification Required

Type	SDP Name	Reference
----	-----	-----
att-field (both session and media level)		
	trafficclass	[this document]

## 7.2 The Traffic Classification Category Registration

This document requests IANA to create a new registry for the traffic Category classes similar to the following table within the Session Description Protocol (SDP) Parameters registry:

Registry Name: "trafficclass" SDP Category Attribute Values

Reference: [this document]

Registration Procedures: Standards-Track document Required

Category Values	Reference
-----	-----
broadcast	[this document]
realtime-interactive	[this document]
multimedia-conferencing	[this document]
multimedia-streaming	[this document]
conversational	[this document]

## 7.3 The Traffic Classification Application Type Registration

This document requests IANA to create a new registry for the traffic application classes similar to the following table within the Session Description Protocol (SDP) Parameters registry:

Registry Name: "trafficclass" SDP Application Attribute Type Values

Reference: [this document]

Registration Procedures: Specification Required

Application Values	Reference
-----	-----
audio	[this document]
video	[this document]
text	[this document]
application-sharing	[this document]
presentation-data	[this document]
whiteboarding	[this document]
instant-messaging	[this document]
gaming	[this document]
remote-desktop	[this document]
telemetry	[this document]
multiplex	[this document]

webcast	[this document]
iptv	[this document]

#### 7.4 The Traffic Classification Adjective Registration

This document requests IANA to create a new registry for the traffic adjective values similar to the following table within the Session Description Protocol (SDP) Parameters registry:

Registry Name: "trafficclass" SDP Adjective Attribute Values  
Reference: [this document]  
Registration Procedures: Specification Required

Adjective Values	Reference
-----	-----
immersive	[this document]
avconf	[this document]
realtime	[this document]
web	[this document]
virtual	[this document]
live	[this document]
surveillance	[this document]
aq:admitted	[this document]
aq:non-admitted	[this document]
aq:partial	[this document]
aq:none	[this document]

#### 7.5 The Traffic Classification Component Mapping

##### 7.5.1 Broadcast Applications and Adjective Combinations

This document requests IANA to create a new registry for the Broadcast category mapping similar to Table 11 in Section 4.5 of this document within the Session Description Protocol (SDP) Parameters registry:

Registry Name: Broadcast Applications and Adjective Combinations  
Table  
Reference: [this document]  
Registration Procedures: TBD

##### 7.5.2 Realtime Interactive Applications and Adjective Combinations

This document requests IANA to create a new registry for the Realtime Interactive category mapping similar to Table 7 in Section 4.3 of this document within the Session Description Protocol (SDP) Parameters registry:

Registry Name: Realtime Interactive Applications and Adjective

## Combinations Table

Reference: [this document]

Registration Procedures: TBD

## 7.5.3 Multimedia Conferencing Applications and Adjective Combinations

This document requests IANA to create a new registry for the Multimedia Conferencing category mapping similar to Table 5 in Section 4.2 of this document within the Session Description Protocol (SDP) Parameters registry:

Registry Name: Multimedia Conferencing Applications and Adjective  
Combinations Table

Reference: [this document]

Registration Procedures: TBD

## 7.5.4 Multimedia-Streaming

This document requests IANA to create a new registry for the Multimedia-Streaming category mapping similar to Table 9 in Section 4.4 of this document within the Session Description Protocol (SDP) Parameters registry:

Registry Name: Multimedia-Streaming Applications and Adjective  
Combinations Table

Reference: [this document]

Registration Procedures: TBD

## 7.5.5 Conversational Applications and Adjective Combinations

This document requests IANA to create a new registry for the conversational category mapping similar to Table 3 in Section 4.1 of this document within the Session Description Protocol (SDP) Parameters registry:

Registry Name: Conversational Applications and Adjective  
Combinations Table

Reference: [this document]

Registration Procedures: TBD

## 8. Acknowledgments

To Dave Oran, Toerless Eckert, Henry Chen, David Benham, David Benham, Mo Zanty, Michael Ramalho, Glen Lavers, Charles Ganzhorn, Paul Kyzivat, Greg Edwards, Charles Eckel, Dan Wing, Cullen Jennings and Peter Saint-Andre for their comments and suggestions.

## 9. References

### 9.1. Normative References

- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997
- [RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997
- [RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers ", RFC 2474, December 1998
- [RFC2872] Y. Bernet, R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", RFC 2872, June 2000
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4080] R. Hancock, G. Karagiannis, J. Loughney, S. Van den Bosch, "Next Steps in Signaling (NSIS): Framework", RFC 4080, June 2005
- [RFC4124] F. Le Faucheur, Ed., " Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering ", RFC 4124, June 2005
- [RFC4566] M. Handley, V. Jacobson, C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006
- [RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5865] F. Baker, J. Polk, M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, May 2010
- [RFC5897] J. Rosenberg, "Identification of Communications Services in the Session Initiation Protocol (SIP)", RFC 5897, June 2010

### 9.2. Informative References

- [RFC4594] J. Babiarz, K. Chan, F Baker, "Configuration Guidelines for Diffserv Service Classes", RFC 4594, August 2006

[ID-RSVP-PROF] J. Polk, S. Dhesikan, "Resource Reservation Protocol (RSVP) Application-ID Profiles for Voice and Video Streams", work in progress, Feb 2013

#### Author's Addresses

James Polk  
3913 Treemont Circle  
Colleyville, Texas, USA  
+1.818.271.3552

mailto: jmpolk@cisco.com

Subha Dhesikan  
170 W Tasman St  
San Jose, CA, USA  
+1.408-902-3351

mailto: sdhesika@cisco.com

Paul E. Jones  
7025 Kit Creek Rd.  
Research Triangle Park, NC, USA  
+1 919 476 2048

mailto: paulej@packetizer.com

#### Appendix - Changes from Previous Versions

##### A.1 From -02 to -03

These are the following changes made between the WG -02 version and the -03 version:

- Rearranged a fair amount of text
- Separated and defined the components into separate subsections.
- built 5 different tables, one per category, that lists within each category - what applications are appropriate as well as what adjectives are appropriate for each application within that category.
- added the 'partial' admission qualifier for those flows that have only part of their respective flow admitted (i.e., CAC'd).

## A.2 From -01 to -02

These are the following changes made between the WG -01 version and the -02 version:

- converged the use of terms 'parent' and 'category' to just 'category' for consistency.
- changed ABNF to reflect extensibility by not having applications and adjectives named in the ABNF, rather have them merely IANA registered.
- merged the qualified and unqualified adjective sections into a single section on adjectives, but allowing some to have a preceding qualifier.
- text clean-up

## A.3 From -00 to -01

These are the following changes made between the WG -00 version and the -01 version:

- removed the non-SDP applications Netflix and VOD
- switched the adjective 'desktop' to 'avconf'
- Labeled each of the figures.
- clarified the differences between Multimedia-Streaming and Broadcast category categories.
- defined Video surveillance
- added the concept of a 'qualified' adjective, and modified the ABNF.
- deleted the idea of a 'cac-class' as a separate component, and made the equivalent a qualified adjective.
- modified the answerer behavior because of the removal of the 'cac-class' component.
- created an IANA registry for qualified adjectives
- general clean-up of the doc.

Did \*not\* do the following in this version:

- add the ability to have more than one trafficclass attribute based on the codec chosen, as feedback indicated this was a bad idea.

- no swap of the Multimedia-Conferencing category with the offered Collaboration category, as doing this did not solve any perceived problems.
- add more to the 'how does this get processed' portion of Section 3. That will come in the next revision.



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 23, 2013

E. Iovov  
Jitsi  
A. Roach  
Mozilla  
February 19, 2013

A Session Initiation Protocol (SIP) usage for Trickle ICE  
draft-iovov-dispatch-sdpfrag-00

## Abstract

This document registers the application/sdpfrag Multipurpose Internet Mail Extensions (MIME) media type. This type is similar to application/sdp, but allows certain subsets of well formed session descriptions, as per the Session Description Protocol (SDP), to be represented instead of requiring a complete SDP session description. The "a=candidate" lines that are incrementally exchanged between Trickle ICE agents are one example usage of the application/sdpfrag.

## Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 23, 2013.

## Copyright Notice

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

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Definition: application/sdpfrag . . . . .	3
4. Security Considerations . . . . .	3
5. Acknowledgements . . . . .	3
6. References . . . . .	4
6.1. Normative References . . . . .	4
6.2. Informative References . . . . .	4
Authors' Addresses . . . . .	4

## 1. Introduction

The application/sdp MIME media type defined in [RFC4566] carries an entire well formed SDP session description. Yet, creating such a description may sometimes require a relatively long time as, for example, would be the case when the Interactive Connectivity Establishment (ICE) [RFC5245] protocol is in use and candidates need to be acquire in different, often time consuming methods. Some applications may therefore choose to use mechanisms like Trickle ICE [I-D.ivov-mmusic-trickle-ice] that would allow them to send initial session descriptions with only readily available information and then exchange candidates only when they become available.

This document does NOT provide a mechanism to segment an SDP session description into multiple pieces for separate transport and later reassemble.

## 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. Definition: application/sdpfrag

A valid application/sdpfrag part is one that could be obtained by starting with some valid SDP session description and deleting any number of lines.

[TODO maybe mention that we can only do frags with the declarative parts of an SDP offer/answer and not with the ones used in negotiations.]

[TODO maybe explain how attributes can be linked to an m line or at least say that this will be defined by usages.]

## 4. Security Considerations

[TODO]

## 5. Acknowledgements

[TODO]

## 6. References

### 6.1. Normative References

- [I-D.ivov-mmusic-trickle-ice]  
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE:  
Incremental Provisioning of Candidates for the Interactive  
Connectivity Establishment (ICE) Protocol",  
draft-ivov-mmusic-trickle-ice-00 (work in progress),  
January 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model  
with Session Description Protocol (SDP)", RFC 3264,  
June 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session  
Description Protocol", RFC 4566, July 2006.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment  
(ICE): A Protocol for Network Address Translator (NAT)  
Traversal for Offer/Answer Protocols", RFC 5245,  
April 2010.

### 6.2. Informative References

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and  
E. Lear, "Address Allocation for Private Internets",  
BCP 5, RFC 1918, February 1996.

## Authors' Addresses

Emil Ivov  
Jitsi  
Strasbourg 67000  
France

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

Adam Roach  
Mozilla  
Dallas, TX  
US

Email: adam@nostrum.com



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 11, 2013

E. Ivov  
Jitsi  
E. Marocco  
Telecom Italia  
C. Holmberg  
Ericsson  
February 7, 2013

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

Abstract

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

This document defines usage semantics for Trickle ICE with SIP.

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 11, 2013.

Copyright Notice

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

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

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

## Table of Contents

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



## 1. Introduction

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

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

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

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

## 2. Terminology

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

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

## 3. Half vs Full Trickle

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

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

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

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

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

#### 4. Encoding and Sending Candidate Information

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

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

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

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

## 5. Info Package

### 5.1. Overall Description

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

### 5.2. Applicability

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

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

### 5.3. INFO Package Name

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

### 5.4. INFO Package Parameters

This document does not define any INFO package parameters.

### 5.5. SIP Option-Tags

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

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

### 5.6. INFO Message Body Parts

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

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

## 6. Example Flows

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

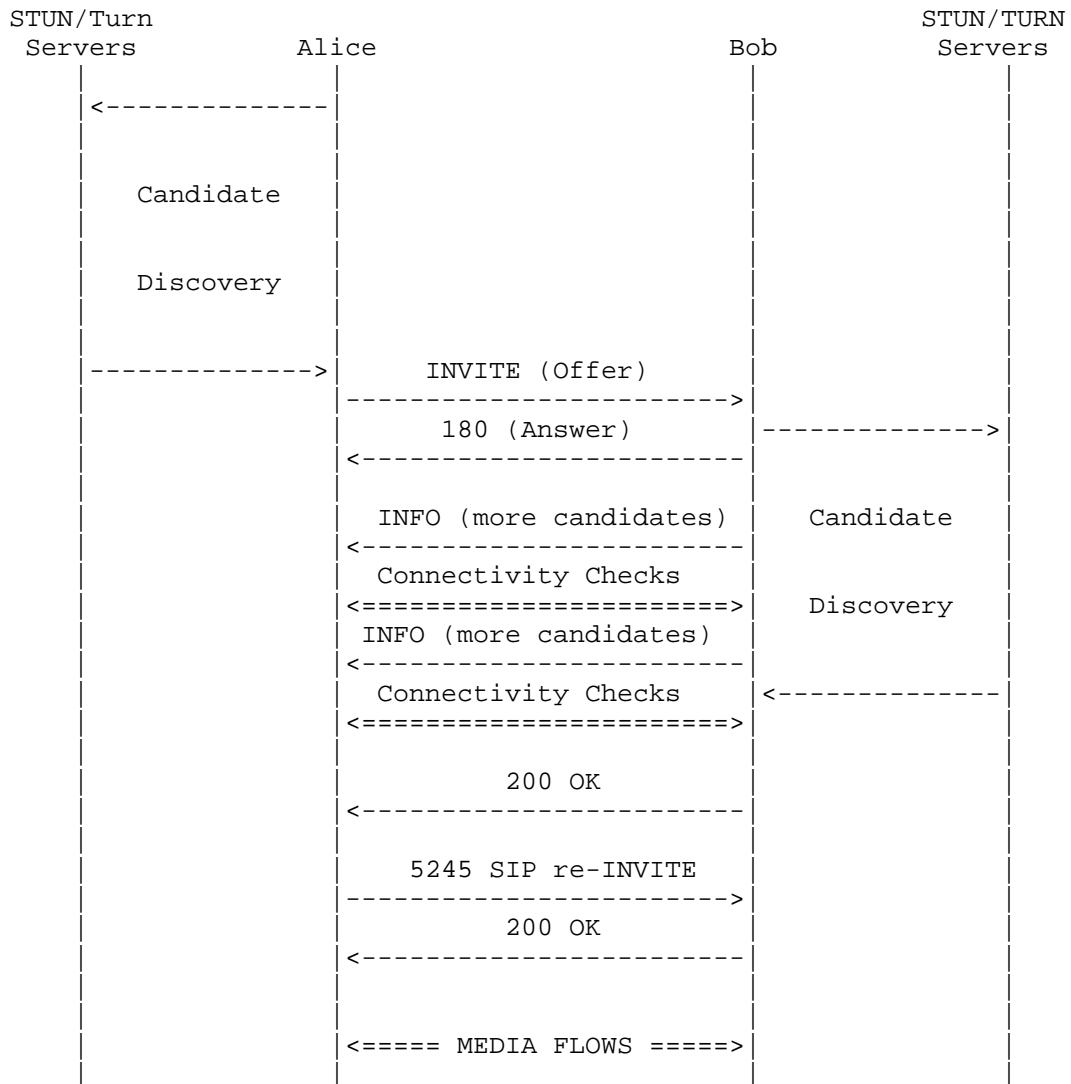


Figure 1: Example

## 7. Security Considerations

[TODO]

## 8. Acknowledgements

[TODO]

## 9. References

### 9.1. Normative References

- [I-D.ivov-mmusic-trickle-ice]  
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE:  
Incremental Provisioning of Candidates for the Interactive  
Connectivity Establishment (ICE) Protocol",  
draft-ivov-mmusic-trickle-ice-00 (work in progress),  
January 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model  
with Session Description Protocol (SDP)", RFC 3264,  
June 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session  
Description Protocol", RFC 4566, July 2006.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment  
(ICE): A Protocol for Network Address Translator (NAT)  
Traversal for Offer/Answer Protocols", RFC 5245,  
April 2010.
- [RFC6086] Holmberg, C., Burger, E., and H. Kaplan, "Session  
Initiation Protocol (SIP) INFO Method and Package  
Framework", RFC 6086, January 2011.

### 9.2. Informative References

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

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

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3388] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.
- [RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

#### Appendix A. Open issues

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

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

Authors' Addresses

Emil Ivov  
Jitsi  
Strasbourg 67000  
France

Phone: +33 6 72 81 15 55  
Email: [emcho@jitsi.org](mailto:emcho@jitsi.org)

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

Email: [enrico.marocco@telecomitalia.it](mailto:enrico.marocco@telecomitalia.it)

Christer Holmberg  
Ericsson  
Hirsalantie 11  
Jorvas 02420  
Finland

Email: [christer.holmberg@ericsson.com](mailto:christer.holmberg@ericsson.com)





Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 1, 2013

E. Iovov  
Jitsi  
E. Rescorla  
RTFM, Inc.  
J. Uberti  
Google  
January 28, 2013

Trickle ICE: Incremental Provisioning of Candidates for the Interactive  
Connectivity Establishment (ICE) Protocol  
draft-iovov-mmusic-trickle-ice-00

## Abstract

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

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

## Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 1, 2013.

## Copyright Notice

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

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

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Incompatibility with Standard ICE . . . . .	4
4. Determining Support for Trickle ICE . . . . .	6
4.1. Unilateral Use of Trickle ICE (Half Trickle) . . . . .	6
5. Sending the Initial Offer . . . . .	7
5.1. Encoding the SDP . . . . .	8
6. Receiving the Initial Offer . . . . .	8
6.1. Sending the Initial Answer . . . . .	9
6.2. Forming check lists and beginning connectivity checks . . . . .	9
6.3. Encoding the SDP . . . . .	10
7. Receiving the Initial Answer . . . . .	10
8. Performing Connectivity Checks . . . . .	10
8.1. Check List and Timer State Updates . . . . .	10
9. Learning and Sending Additional Local Candidates . . . . .	11
9.1. Encoding the SDP for Additional Candidates . . . . .	12
9.2. Announcing End of Candidates . . . . .	13
9.3. Receiving an End Of Candidates Notification . . . . .	14
10. Receiving Additional Remote Candidates . . . . .	14
11. Concluding ICE Processing . . . . .	14
12. Subsequent Offer/Answer Exchanges . . . . .	14
13. Interaction with ICE Lite . . . . .	15
14. Example Flow . . . . .	15
15. Security Considerations . . . . .	16
16. Acknowledgements . . . . .	16
17. References . . . . .	16
17.1. Normative References . . . . .	16
17.2. Informative References . . . . .	16
Appendix A. Open issues . . . . .	17
A.1. MID/Stream Indices in SDP . . . . .	18
A.2. ICE Lite and Candidate Signalling . . . . .	18
A.3. Handling new candidates after check completion . . . . .	18
Appendix B. Changes From Earlier Versions . . . . .	18
B.1. Changes From draft-rescorla-01 . . . . .	18
B.2. Changes From draft-rescorla-00 . . . . .	19
Authors' Addresses . . . . .	19

## 1. Introduction

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

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

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

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

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

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

In addition to the basics of trickle ICE, this document also describes how support for trickle ICE needs to be discovered, how

regular ICE processing needs to be modified when building and updating check lists, and how trickle ICE implementations should interoperate with agents that only implement [RFC5245] processing.

This specification does not define usage of trickle ICE with any specific signalling protocol, contrary to [RFC5245] which contains a usage for ICE w/ SIP. Such usages would have to be specified in separate documents.

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

## 2. Terminology

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

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

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

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

## 3. Incompatibility with Standard ICE

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

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

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

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

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

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

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

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

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

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

#### 4. Determining Support for Trickle ICE

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

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

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

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

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

##### 4.1. Unilateral Use of Trickle ICE (Half Trickle)

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

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

name of the mechanism).

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

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

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

## 5. Sending the Initial Offer

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

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

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

checks.

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

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

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

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

#### 5.1. Encoding the SDP

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

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

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

As mentioned earlier in this section, Offers and Answers can contain any set of candidates, which means that a trickle ICE session description MAY contain no candidates at all. In such cases the agent would still need to place an address in the "c=" line(s). If the use of a host address there is undesirable (e.g. for privacy reasons), the agent MAY set the connection address to 0.0.0.0. In this case it MUST also set the port number to 1. There is no need to include a fictitious 0.0.0.0 candidate when doing so.

#### 6. Receiving the Initial Offer

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



It is worth pointing out that in case support for trickle ICE is confirmed, an agent will automatically assume support for vanilla ICE as well even if the support verification procedure in [RFC5245] indicates otherwise. Specifically, such verification would indicate lack of support when the offer contains no candidates. The 0.0.0.0 address present in the c= line in that case would not "appear in a candidate attribute". Obviously, a fallback to [RFC3264] is not required when this happens.

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

#### 6.1. Sending the Initial Answer

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

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

#### 6.2. Forming check lists and beginning connectivity checks

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

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

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

first non-empty checklist.

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

### 6.3. Encoding the SDP

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

## 7. Receiving the Initial Answer

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

## 8. Performing Connectivity Checks

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

### 8.1. Check List and Timer State Updates

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

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

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

- o all candidate harvesters have completed and the agent is not expecting to learn any new candidates;

- o the remote agent has sent an end-of-candidates indication for that check list as described in Section 9.2.

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

## 9. Learning and Sending Additional Local Candidates

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

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

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

Otherwise, if the agent has already learned of one or more remote candidates for this stream and component, it will begin pairing the new local candidates with them and adding the pairs to the existing check lists according to their priority. Forming candidate pairs will work the way it is described by the vanilla ICE specification. Actually adding the new pair to a check list however, will happen according to the rules described below.

If the new pair's local candidate is server reflexive, the server reflexive candidate MUST be replaced by its base before adding the

pair to the list. Once this is done, the agent examines the check list looking for another pair that would be redundant with the new one. If such a pair exists and its state is:

Succeeded: the newly formed pair is ignored.

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

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

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

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

- o if this check list is Frozen then the new pair will also be assigned a Frozen state.
- o else if the check list is Active and it is either empty or contains only candidates in the Succeeded and Failed states, then the new pair's state is set to Waiting.
- o else if the check list is non-empty and Active, then the new pair state will be set to

Frozen: if there is at least one pair in the list whose foundation matches the one in the new pair and whose state is neither Succeeded nor Failed (eventually the new pair will get unfrozen after the the on-going check for the existing pair concludes);

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

### 9.1. Encoding the SDP for Additional Candidates

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

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

Given that such lines do not provide a relationship between the candidate and the m line that it relates to, signalling protocols using trickle ICE MUST establish that relation themselves. This can

be done in one of two ways: using the m-line index, or an MID [RFC3388]. The m-line index is a zero-based index, referring to the Nth m-line in the SDP. The MID uses "media stream identification", as defined in [RFC3388], to identify the m-line. When creating candidate lines usages of trickle ICE SHOULD use the MID if possible, or the m-line index if not. Agents MUST NOT send individual candidates any prior to generating the corresponding SDP session description.

The exact means of transporting additional candidates to a remote agent is left to the protocols using trickle ICE. It is important to note, however, that these candidate exchanges are not part of the offer/answer model. [TODO: or should we pick one of the two? Is there any reason not to mandate MID?]

## 9.2. Announcing End of Candidates

Once all candidate harvesters for a specific media stream complete, or expire, the agent MUST generate an "end-of-candidates" indication for that stream and send it to the remote agent via the signalling channel. The SDP representation for end-of-candidates following form:

a=end-of-candidates

The end-of-candidates notifications can be sent as part of an offer, which would typically be the case with half trickle initial offers, they can accompany the last candidate an agent can send for a stream, and they can also be sent alone (e.g. after STUN Binding requests or TURN Allocate requests to a server timeout and the agent has no other active harvesters).

Receiving an end-of-candidates notification allows an agent to update check list states and, in case valid pairs do not exist for every component in every media stream, determine that ICE processing has failed.

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

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

support trickle ICE. (TODO: can this be useful?)

Once the agent sends the end-of-candidates event, it will update the state of the corresponding check list as explained in section Section 8.1

### 9.3. Receiving an End Of Candidates Notification

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

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

## 10. Receiving Additional Remote Candidates

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

Otherwise, the new candidates are used for forming candidate pairs with the pool of local candidates.

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

## 11. Concluding ICE Processing

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

## 12. Subsequent Offer/Answer Exchanges

Either agent MAY generate a subsequent offer at any time allowed by [RFC3264]. When this happens agents will use [RFC5245] semantics to determine whether or not the new offer requires an ICE restart. If

this is the case then agents would perform trickle ICE as they would in an initial offer/answer exchange.

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

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

### 13. Interaction with ICE Lite

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

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

### 14. Example Flow

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

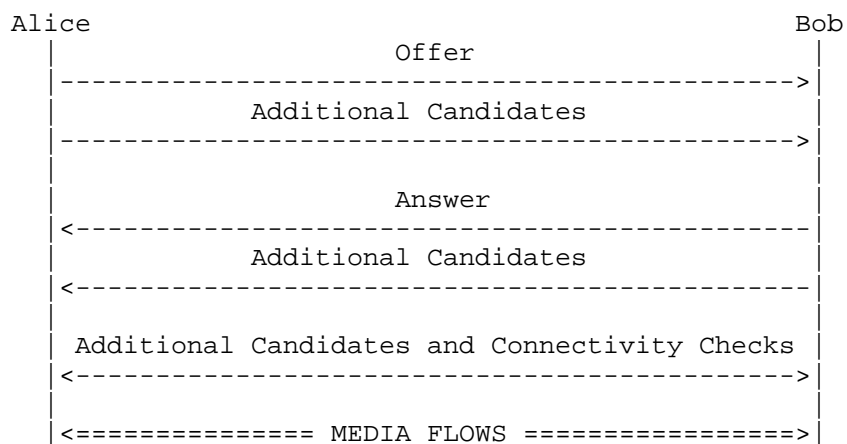


Figure 1: Example

## 15. Security Considerations

[TODO]

## 16. Acknowledgements

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

## 17. References

## 17.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

## 17.2. Informative References

- [I-D.keranen-mmusic-ice-address-selection] Keraenen, A. and J. Arkko, "Update on Candidate Address Selection for Interactive Connectivity Establishment (ICE)", draft-keranen-mmusic-ice-address-selection-01 (work in progress), July 2012.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E.



- Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3388] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.
- [RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [XEP-0030] Hildebrand, J., Millard, P., Eatmon, R., and P. Saint-Andre, "XEP-0030: Service Discovery", XEP XEP-0030, June 2008.
- [XEP-0115] Hildebrand, J., Saint-Andre, P., Troncon, R., and J. Konieczny, "XEP-0115: Entity Capabilities", XEP XEP-0115, February 2008.
- [XEP-0176] Beda, J., Ludwig, S., Saint-Andre, P., Hildebrand, J., Egan, S., and R. McQueen, "XEP-0176: Jingle ICE-UDP Transport Method", XEP XEP-0176, June 2009.
- [XEP-0278] Camargo, T., "XEP-0278: Jingle Relay Nodes", XEP XEP-0278, June 2011.

#### Appendix A. Open issues

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

### A.1. MID/Stream Indices in SDP

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

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

### A.2. ICE Lite and Candidate Signalling

ICE Lite implementations of Trickle ICE do not necessarily need to receive candidates from their peers: it would be enough for them to start getting checks and discover all candidates as peer reflexive. This would allow to reduce signalling traffic but could introduce other issues.

### A.3. Handling new candidates after check completion

Basically, do any new candidates discovered after ICE completes (e.g. from a new interface that comes up) need an ICE restart, or not?

## Appendix B. Changes From Earlier Versions

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

### B.1. Changes From draft-rescorla-01

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

they want.

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

#### B.2. Changes From draft-rescorla-00

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

#### Authors' Addresses

Emil Ivov  
Jitsi  
Strasbourg 67000  
France

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

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

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

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

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



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 17, 2013

C. Jennings  
Cisco  
J. Uberti  
Google  
E. Rescorla  
Mozilla  
February 13, 2013

Requirements from various WG for MMUSIC  
draft-jennings-mmusic-media-req-00

Abstract

This document outlines some of the requirements driving various consideration related to multiplexing in the MMUSIC working group to meet the needs of WebRTC, CLUE, and other working groups.

This document is only meant to be used to help drive the discussion of solutions and is not intended to become an RFC.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 17, 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Requirements . . . . .	3
4. Non-Requirements . . . . .	4
5. IANA Considerations . . . . .	4
6. Security Considerations . . . . .	4
7. Acknowledgements . . . . .	4
8. References . . . . .	4
8.1. Normative References . . . . .	4
8.2. Informative References . . . . .	5
Authors' Addresses . . . . .	5

## 1. Introduction

For the past several meetings, there has been discussions around various mechanism to reduce the number of UDP ports needed by applications for RTP. This document attempts to capture some of the requirements that are important in selecting the solution for how to represent the SDP to negotiate the RTP media that is using reduced ports.

## 2. Terminology

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

## 3. Requirements

This section covers the requirements from various WG for setting up media. Obviously it does not try and cover all the requirements but it tries to cover a set that seem relevant to decisions around multiplexing onto few UDP ports.

### High Priority Requirements:

1. Support many media flows but minimize the number of transport flows. For instance, all media flows--or perhaps all media flows of a given type--might be multiplexed over a single transport flow.
2. Be able to successfully negotiate media with both legacy SIP devices and new devices (whether SIP or RTCWEB) with a single offer/answer exchange. If both endpoints support multiplexed media, then multiplexing should be negotiated. Otherwise, non-multiplexed media should be used. In many cases, each endpoints will have no prior knowledge of capabilities of the other endpoint.

### Other Requirements:

1. Need a uniform way to allow specifications of new SDP parameters to easily explain any implications that multiplexing has on the new parameters in that specification.
2. Allow different sources (E.g. cameras) to use different codecs. For example, if one camera had hardware encoders for VP8 while



another had encoders for H.264, the device may wish to negotiate different codecs.

3. Be able to to independently set parameters such as resolution bandwidth, independently for each RTCWeb Track, preferably even when they are all multiplexed over the same transport flow.
4. Be able to identify the RTCWeb tracks with an identifier that is stable over the duration of the session. More information can be found in [I-D.alvestrand-mmusic-msid].

#### 4. Non-Requirements

Some items are not a major goal. If methods are found that work for these as well, that is great, but they are not a priority item.

1. Working with SIP proxies or B2BUA that are not compliant with the standards. The reason for this is it is just not possible to design for every possible thing that does not do what the standards require.

#### 5. IANA Considerations

This document makes no request of IANA.

#### 6. Security Considerations

These requirements have no additional security considerations other than those captured in [I-D.ietf-rtcweb-security-arch].

#### 7. Acknowledgements

Thanks to ...

#### 8. References

##### 8.1. Normative References

[I-D.ietf-rtcweb-security-arch]  
Rescorla, E., "RTCWEB Security Architecture",  
draft-ietf-rtcweb-security-arch-06 (work in progress),  
October 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

## 8.2. Informative References

[I-D.alvestrand-mmusic-msid]  
Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol",  
draft-alvestrand-mmusic-msid-02 (work in progress),  
December 2012.

## Authors' Addresses

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

Email: fluffy@iii.ca

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

Email: justin@uberti.name

Eric Rescorla  
Mozilla

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



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 29, 2013

C. Jennings  
Cisco  
February 25, 2013

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

Abstract

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

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

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

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

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

## Table of Contents

1. Overview . . . . .	4
2. Terminology . . . . .	6
3. Requirements . . . . .	7
4. Background/Solution Overview . . . . .	9
5. Overall Design . . . . .	11
6. Example Mappings . . . . .	12
6.1. One Audio, One Video, No bundle/multiplexing . . . . .	12
6.2. One Audio, One Video, Bundle/multiplexing . . . . .	12
6.3. One Audio, One Video, Simulcast, Bundle/multiplexing . . . . .	12
6.4. One Audio, One Video, Bundle/multiplexing, Lip-Sync . . . . .	12
6.5. One Audio, One Active Video, 5 Thumbnails, Bundle/multiplexing . . . . .	12
6.6. One Audio, One Active Video, 5 Thumbnails, Main Speaker Lip-Sync, Bundle/multiplexing . . . . .	13
7. Solutions . . . . .	14
7.1. Correlation and Multiplexing . . . . .	15
7.2. Multiple Render . . . . .	18
7.2.1. Complex Multi Render Example . . . . .	18
7.3. Dirty Little Secrets . . . . .	22
7.4. Open Issues . . . . .	22
7.5. Confusions . . . . .	22
8. Examples . . . . .	23
9. Tasks . . . . .	27
10. Security Considerations . . . . .	28
11. IANA Considerations . . . . .	29
12. Acknowledgments . . . . .	30
13. Open Issues . . . . .	31
14. Existing SDP . . . . .	32
14.1. Multiple Encodings . . . . .	32
14.2. Forward Error Correction . . . . .	33
14.3. Same Video Codec With Different Settings . . . . .	33
14.4. Different Video Codecs With Different Resolutions Formats . . . . .	34
14.5. Lip Sync Group . . . . .	34
14.6. BFCP . . . . .	34
14.7. Retransmission . . . . .	35
14.8. Layered coding dependency . . . . .	37
14.9. SSRC Signaling . . . . .	38
14.10. Content Signaling . . . . .	39
15. References . . . . .	40
15.1. Normative References . . . . .	40
15.2. Informative References . . . . .	40
Author's Address . . . . .	42

## 1. Overview

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

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

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

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

This does results in lots of m lines but all the alternatives designs resulted in an roughly equivalent number of SSRC lines with a possibility of redefining most of the media level attributes. So it's really hard to see the big benefits defining something new over what we have. One of the concerns about this approach is the time to collect all the ICE candidates needed for the initial offer. Section 7.2.1 provides mitigations to reduce the number of ports needed to be the same as an alternative SSRC based design. This assumes that it is perfectly feasible to transport SDP that much larger than a single MTU. The SIP [RFC3261] usage of SDP has successfully passed over this long ago. In the cases where the SDP

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



## 2. Terminology

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

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

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

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

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

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

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

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

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

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

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

### 3. Requirements

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

- o Devices with multiple audio and/or video sources
- o Devices that display multiple streams of video and/or render multiple streams of audio
- o Simulcast, wherein a video from a single camera is sent in a few independent video streams typically at different resolutions and frame rates.
- o Layered Codec such as H.264 SVC
- o One way media flows and bi-directional media flows
- o Support asymmetry, i.e. to send a different number of type of media streams that you receive.
- o Mapping W3C PeerConnection (PC) aspects into SDP and RTP. It is important that the SDP be descriptive enough that both sides can get the same view of various identifiers for PC-Tracks, PC-Streams and their relationships.
- o Support of Interactive Connectivity Establishment (ICE) [RFC5245]
- o Support of multiplexing multiple media flows, possible of different media types, on same 5-tuple.
- o Synchronization - It needs to be clear how implementations deal with synchronization, in particular usages of both CNAME and LS group. The sender needs be able to indicate which Media Flows are intended to be synchronized and which are not.
- o Redundant codings - The ability to send some media, such as the audio from a microphone, multiple times. For example it may be sent with a high quality wideband codec and a low bandwidth codec. If packets are lost from the high bandwidth steam, the low bandwidth stream can be used to fill in the missing gaps of audio. This is very similar to simulcast.

- o Forward Error Correction - Support for various RTP FEC schemes.
- o RSVP QoS - Ability to signal various QoS mechanism such Single Reservation Flow (SRF) group
- o Desegregated Media (FID group) - There is a growing desire to deal with endpoints that are distributed - for example a video phone where the incoming video is displayed on the an IP TV but the outgoing video comes from a tablet computer. This results in situations where the SDP sets up a session with not all the media transmitted to a single IP address.
- o In flight change of codec: Support for system that can negotiate the uses of more than one codec for a given media flow and then the sender can arbitrarily switch between them when they are sending but they only send with one codec as at time.
- o Distinguish simulcast (e.g. multiple encoding of same source) from multiple different sources
- o Support for Sequential and Parallel forking at the SIP level
- o Support for Early Media
- o Conferencing environments with Transcoding MCU that decodes/mixes/ recodes the media
- o Conferencing environments with Switching MCU where the MCU mucks the header information of the media and do not decode/recode all the media

#### 4. Background/Solution Overview

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

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

Offer

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

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

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

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

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

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

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

m-line multiplexing (bundle):

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

multi-render:

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

Obviously, this strategy does not scale to large numbers. For instance, consider the case where we want to be able to transmit 35 video thumbnails (this is large, but not insane). In the model described above, each of these flows would need its own m-line and its own set of codecs. If each side supports three separate codecs (e.g., H.261, H.263, and VP8), then we have just consumed 105 payload types, which exceeds the available dynamic payload space.

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

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

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

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

## 5. Overall Design

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

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

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

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

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

## 6. Example Mappings

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

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

```
Microphone --> m=audio --> Speaker > 5-Tuple
Camera      --> m=video --> Window  > 5-Tuple
```

### 6.2. One Audio, One Video, Bundle/multiplexing

```
Microphone --> m=audio --> Speaker \
Camera      --> m=video --> Window  / > 5-Tuple
```

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

```
Microphone --> m=audio --> Speaker \
Camera      +-> m=video -\         | > 5-Tuple
              |         ?-> Window |
              +-> m=video -/         /
```

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

```
Microphone --> m=audio --> Speaker \
Camera      --> m=video --> Window  / > 5-Tuple, Lip-Sync
                                           group
```

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

```
Microphone --> m=audio --> Speaker \
Camera      --> m=video --> Window  | > 5-Tuple
Camera      --> m=video --> 5 Small Windows |
Camera      --> a=multi-render:5             |
...                                           /
```

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

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

```

Microphone -->    m=audio    -->    Speaker    \    \    Lip-sync
                                           |    > group
Camera      -->    m=video    -->    Window      |    /
                                           > 5-Tuple
Camera      -->    m=video    -->    5 Small Windows |
Camera      -->    a=multi-render:5                  |
...                                                  /

```



## 7. Solutions

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

### Approach:

1. If a system wants to offer to send two sources, such as two camera, it **MUST** use a separate m-block for each source. The means that each PC-Track corresponds to one or more m-blocks.
2. In cases such as FEC, simulcast, SVC, each repair stream, layer, or simulcast media flow will get an m-block per media flow.
3. If a systems wants to receive two streams of video to display in two different windows or screens, it **MUST** use separate m-blocks for each unless explicitly signaled to be otherwise (see Section 7.2).
4. Unless explicitly signaled otherwise (see Section 7.2), if a given m-line receives media from multiple SSRCs, only media from the most recently received SSRC **SHOULD** be rendered and other SSRC **SHOULD NOT** and if it is video it **SHOULD** be rendered in the same window or screen.
5. If a camera is sending simulcast video and three resolutions, each resolution **MUST** get its own m-block and all the three m-blocks will be grouped. A new SDP group will be defined for this.
6. If a camera is using a layered codec with three layers, there **MUST** be an m-block for each, and they will be grouped using standard SDP for grouping layers.
7. To aid in synchronized playback, there is exactly one, and only one, LS group for each PC-Stream. All the m-blocks for all the PC-Tracks in a given PC-Stream are synchronized so they are all put in one LS group. All the PC-Tracks in a given PC-Stream have the same CNAME. If a PC-Track appears in more than one PC-Stream, then all the PC-Streams with that PC-Track **MUST** have the same CNAME.
8. One way media **MUST** use the sendonly or recvonly attributes.

9. Media lines that are not currently in use but may be used later, so that the resources need to be kept allocated, SHOULD use the inactive attribute.
10. If an m-line will not be used, or it is rejected, it MUST have its port set to zero.
11. If a video switching MCU produces a virtual "active speaker" media flow, that media flow should have its own SSRC but include the SSRC of the current speaker's video in the CSRC packets it produces.
12. For each PC-Track, the W3C API MUST provide a way to set and read the CSRC list, set and read the content RFC 4574 "label", and read the SSRC of last packet received on a PC-Track.
13. The W3C API should have a constraint or API method to allow a PC-Stream to indicate the number of multi-render video streams it can accept. Each time a new stream is received up to the maximum, a new PC-Track will be created.
14. Applications MAY signal all the SSRC they intend to send using RFC 5576, but receivers need to be careful in their usage of the SSRC in signaling, as the SSRC can change when there is a collision and it takes time before that will be updated in signaling.
15. Applications can get out of band "roster information" that maps the names of various speakers or other information to the MSID and/or SSRCs that a user is using
16. Applications MAY use RFC 4574 content labels to indicate the purpose of the video. The additional content types, main-left and main-right, need to be added to support two- and three-screen systems.
17. The CLUE WG might want to consider SDP to signal the 3D location and field of view parameters for captures and renderers.
18. The W3C API allows a "label" to be set for the PC-Track. This MUST be mapped to the SDP label attribute.

#### 7.1. Correlation and Multiplexing

The port number that RTP is received on provides the primary mechanism for correlating it to the correct m-line. However, when the port does not uniquely male the RTP packet to the correct m-block (such as in multiplexing and other cases), the next thing that can be

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

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

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

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

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

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

```
a=msid:12345123451234512345 s:1 f:3
```

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

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

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

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

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

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

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

## 7.2. Multiple Render

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

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

Offer

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

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

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

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

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

### 7.2.1. Complex Multi Render Example

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

## Offer

```
v=0
o=alice 20519 0 IN IP4 0.0.0.0
s=ULP FEC
t=0 0
a=ice-ufrag:074c6550
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:
c9:c7:70:9d:1f:66:79:a8:07
c= IN IP4 24.23.204.141
a=group:BUNDLE vid1 vid2 vid3
a=group:FEC vid1 vid2
a=group:SIMULCAST vid1 vid3

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

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

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

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

Answer

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

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

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

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

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

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

Answer

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

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

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

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



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

### 7.3. Dirty Little Secrets

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

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

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

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

### 7.4. Open Issues

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

### 7.5. Confusions

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

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

## 8. Examples

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

## Offer

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

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

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

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

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

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

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

## Offer

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

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

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

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

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

## Answer

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

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

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

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

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

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

Offer

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

m=audio 49170 RTP/AVP 99
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d
:f7:c9:c7:70:9d:1f:66:79:a8:07
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQXlcfHAWJSoj|2^20|1:32
a=rtpmap:99 iLBC/8000

m=video 51372 RTP/AVP 96
a=fingerprint:sha-1 92:81:49:83:4a:23:0a:0f:1f:9d:f7:
c0:c7:70:9d:1f:66:79:a8:07
a=crypto:1 AES_CM_128_HMAC_SHA1_32
inline:NzB4dlBINUAvLEw6UzF3WSJ+PSdFcGdUJShpXlZj|2^20|1:32
a=rtpmap:96 H261/90000
```

Answer

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

m=audio 49172 RTP/AVP 99
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQXlcfHAWJSoj|2^20|1:32
a=rtpmap:99 iLBC/8000

m=video 51374 RTP/AVP 96
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQXlcfHAWJSoj|2^20|1:32
a=rtpmap:96 H261/90000
```

## 9. Tasks

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

### Tasks:

1. Extend the W3C API to be able to set and read the CSRC list for a PC-Track.
2. Extend the W3C API to be able to read SSRC of last RTP packed received.
3. Write an RTP Header Extension draft to carry the MSID.
4. Fix up MSID draft to align with this proposal.
5. Write a draft to add left, right to the SDP content attribute. Add the stuff to the W3C API to read and write this on a track.
6. Write a draft on SDP "SIMULCAST" group to signal multiple m-block as are simulcast of same video content.
7. Complete the bundle draft.
8. Provide guidance for ways to use SDP for reduced glare when adding of one way media streams.
9. Write a draft defining the multi render attribute.
10. Change W3C API to say that a PC-Track can be in only one PeerConnection or make an object inside the PeerConnection for each track in the PC that can be used to set constraints and settings and get information related to the RTP flow.
11. Sort out how to tell a PC-Track, particularly one meant for receiving information, that it can do simulcast, layered coding, RTX, FEC, etc.

## 10. Security Considerations

TBD

## 11. IANA Considerations

This document requires no actions from IANA.



## 12. Acknowledgments

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

### 13. Open Issues

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

## 14. Existing SDP

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

### 14.1. Multiple Encodings

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

Offer

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

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

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

Answer

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

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

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

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

#### 14.2. Forward Error Correction

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

Offer

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

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

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

#### 14.3. Same Video Codec With Different Settings

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

Offer

```
v=0

m=video 49170 RTP/AVP 100 99 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E; packetization-mode=0;
sprop-parameter-sets=Z0IACpZTBmI,aMljiA==
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=42A01E; packetization-mode=1;
sprop-parameter-sets=Z0IACpZTBmI,aMljiA==
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42A01E; packetization-mode=2;
sprop-parameter-sets=Z0IACpZTBmI,aMljiA==;
sprop-interleaving-depth=45; sprop-deint-buf-req=64000;
sprop-init-buf-time=102478; deint-buf-cap=128000
```

#### 14.4. Different Video Codecs With Different Resolutions Formats

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

Offer

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

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

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

#### 14.5. Lip Sync Group

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

Offer

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

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

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

#### 14.6. BFCP

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

## Offer

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

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

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

## Answer

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

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

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

## 14.7. Retransmission

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

## Offer

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

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

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

m=video 49174 RTP/AVPF 98
a=rtpmap:98 MP4V-ES/90000
a=rtcp-fb:98 nack
a=fmtp:98 profile-level-id=8;config=01010000012000884006682C209\
0A21F
a=mid:3

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

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

## SDP

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

m=video 49170 RTP/AVPF 96 97
a=rtpmap:96 MP4V-ES/90000
a=rtcp-fb:96 nack
a=fmtp:96 profile-level-id=8;config=010100000012000884006682C209\
0A21F
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96;rtx-time=3000
```

## 14.8. Layered coding dependency

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



## Offer

```
a=group:DDP L1 L2 L3

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

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

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

## 14.9. SSRC Signaling

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

## Offer

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

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

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

#### 14.10. Content Signaling

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

Offer

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

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

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

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

## 15. References

### 15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

### 15.2. Informative References

- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4583] Camarillo, G., "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", RFC 4583, November 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4756] Li, A., "Forward Error Correction Grouping Semantics in Session Description Protocol", RFC 4756, November 2006.
- [RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.
- [webrtc-api] Bergkvist, Burnett, Jennings, Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", October 2011.
- Available at  
<http://dev.w3.org/2011/webrtc/editor/webrtc.html>

Author's Address

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

Email: fluffy@iii.ca



MMUSIC  
Internet-Draft  
Obsoletes: 5245 (if approved)  
Intended status: Standards Track  
Expires: August 29, 2013

A. Keranen  
Ericsson  
J. Rosenberg  
jdrosen.net  
February 25, 2013

Interactive Connectivity Establishment (ICE): A Protocol for Network  
Address Translator (NAT) Traversal for Offer/Answer Protocols  
draft-keranen-mmusic-rfc5245bis-01

## 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 August 29, 2013.

## Copyright Notice

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

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

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

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 . . . . .	20
4.1.1.3. Computing Foundations . . . . .	22
4.1.1.4. Keeping Candidates Alive . . . . .	22
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 . . . . .	28
5.3. Gathering Candidates . . . . .	29
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 . . . . .	49
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 . . . . .	52
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 . . . . .	56
11.	Extensibility Considerations . . . . .	57
12.	Setting Ta and RTO . . . . .	58
12.1.	RTP Media Streams . . . . .	58
12.2.	Non-RTP Sessions . . . . .	60
13.	Example . . . . .	60
14.	Security Considerations . . . . .	65
14.1.	Attacks on Connectivity Checks . . . . .	65
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 . . . . .	71
16.2.2. Gathering and Connectivity Checks . . . . .	72
16.2.3. Keepalives . . . . .	72
16.3. ICE and ICE-lite . . . . .	73
16.4. Troubleshooting and Performance Management . . . . .	73
16.5. Endpoint Configuration . . . . .	73
17. IANA Considerations . . . . .	74
17.1. STUN Attributes . . . . .	74
17.2. STUN Error Responses . . . . .	74
18. IAB Considerations . . . . .	74
18.1. Problem Definition . . . . .	74
18.2. Exit Strategy . . . . .	75
18.3. Brittleness Introduced by ICE . . . . .	75
18.4. Requirements for a Long-Term Solution . . . . .	76
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 . . . . .	80
Appendix B. Design Motivations . . . . .	81
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.

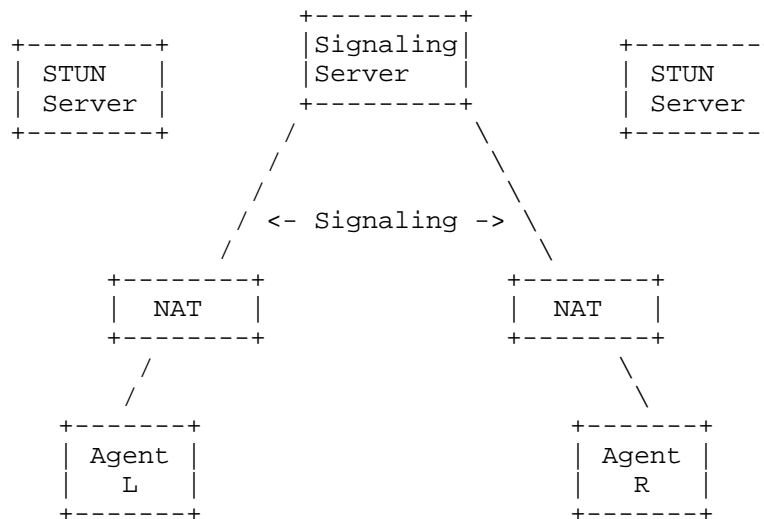


Figure 1: ICE Deployment Scenario

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.

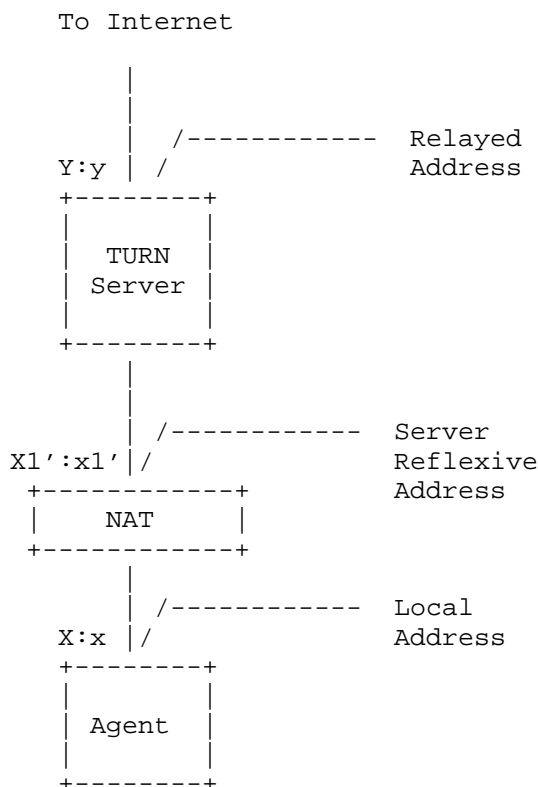


Figure 2: Candidate Relationships

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:

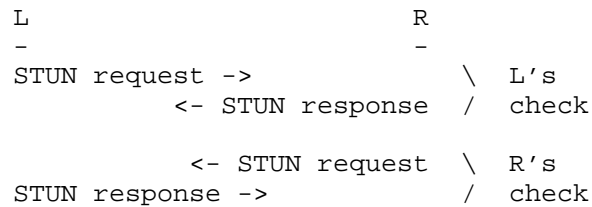


Figure 3: Basic Connectivity Check

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. It 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.

###### 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].

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.

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.

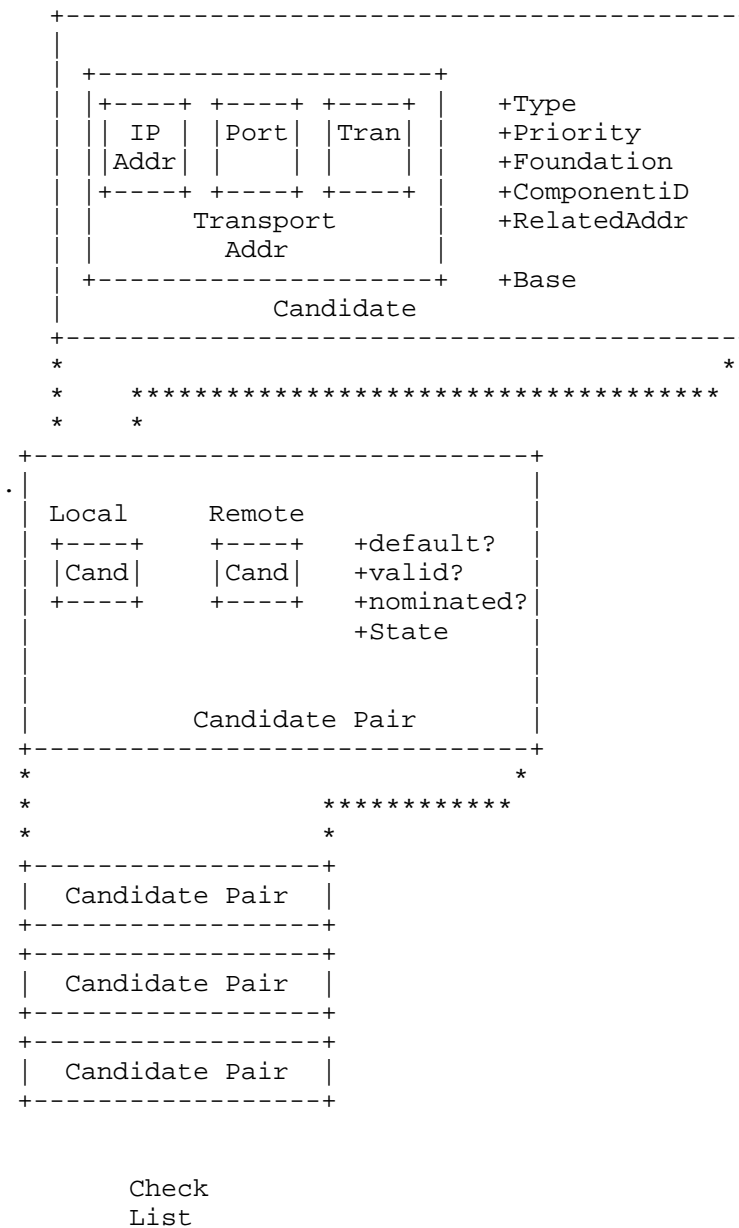


Figure 6: Conceptual Diagram of a Check List

#### 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.



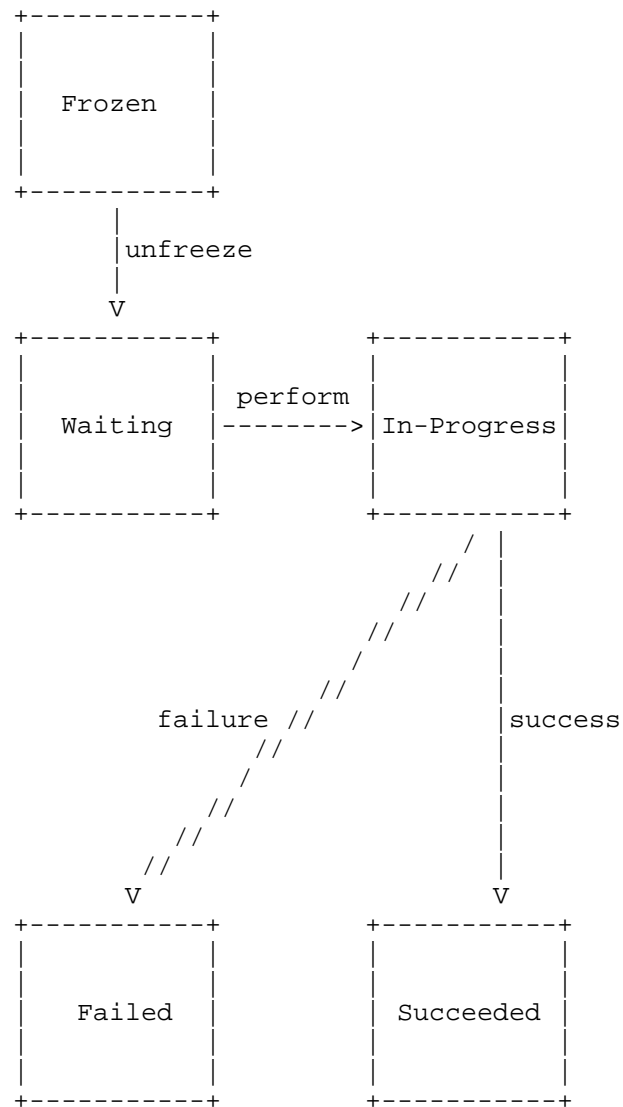


Figure 7: Pair State FSM

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 a short-term credential to authenticate the request and perform a message integrity check. 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  $T_r$  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:

$Ta_i = (stun\_packet\_size / rtp\_packet\_size) * rtp\_ptime$

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



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

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

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:

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

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,  $T_a$  and  $RTO$  revert to more conservative values.  $T_a$  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:

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

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:

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

## 13. Example

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

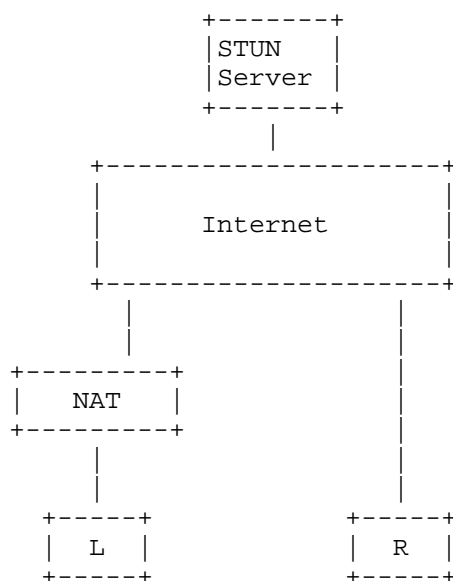


Figure 8: Example Topology

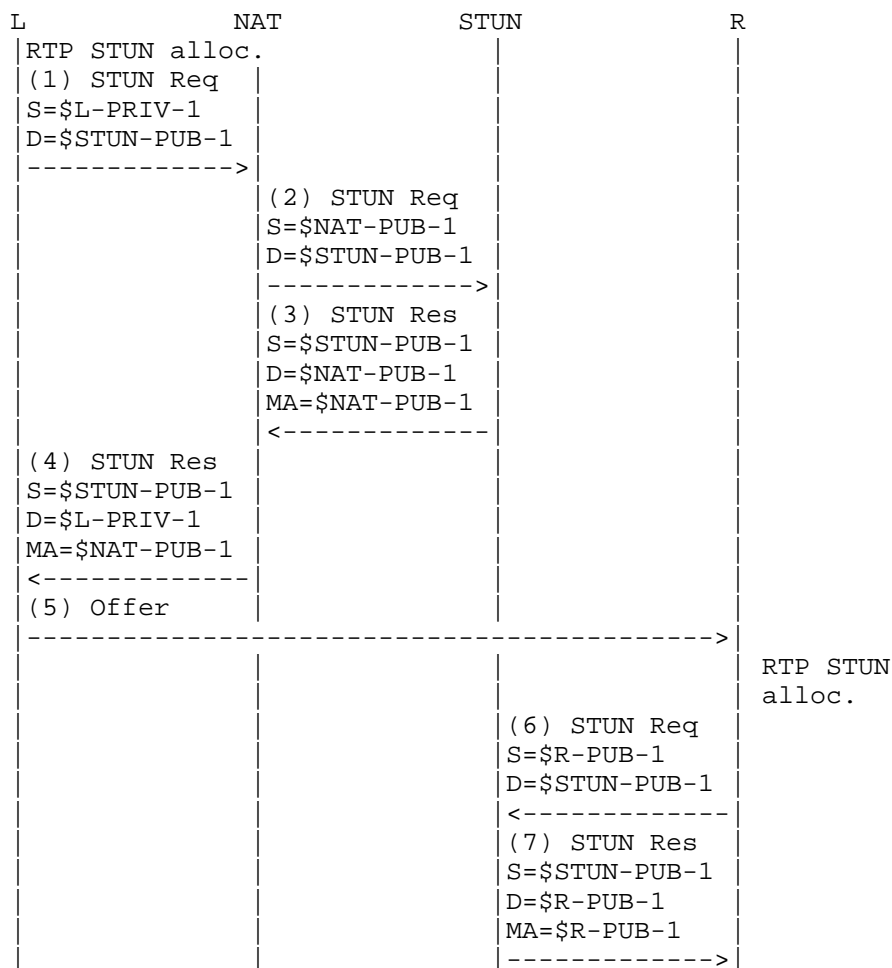
Two agents, L and R, are using ICE. Both are full-mode ICE implementations and use aggressive nomination when they are controlling. Both agents have a single IPv4 address. For agent L, it is 10.0.1.1 in private address space [RFC1918], and for agent R, 192.0.2.1 on the public Internet. Both are configured 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.



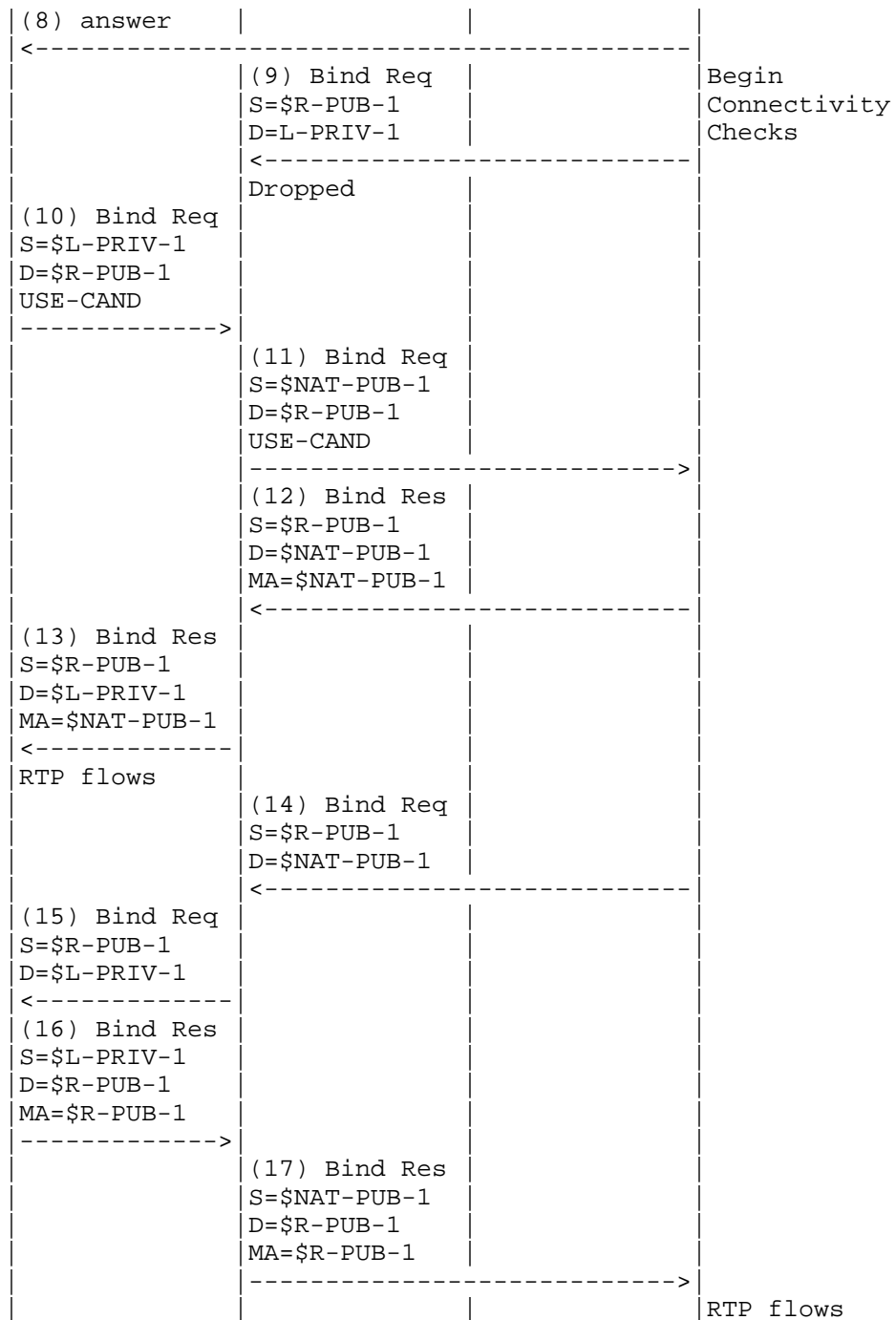


Figure 9: Example Flow

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  $N \times 1.7$  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  $N \times 3.2$  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 removed 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.

## 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", STD 64, 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.
- [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.
- [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, R., Karrenberg, D., 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.

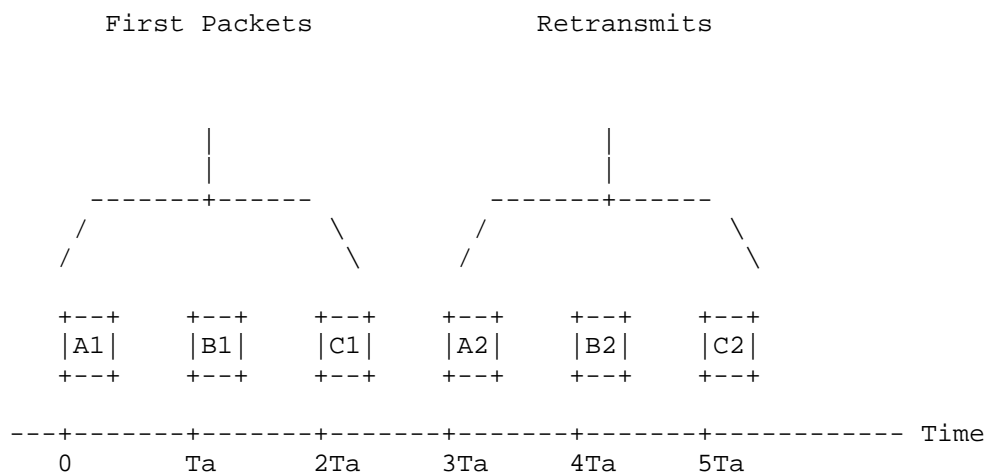
#### B.1. Pacing of STUN Transactions

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 3Ta.

Subsequent retransmits after the first will occur even less frequently than Ta 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:

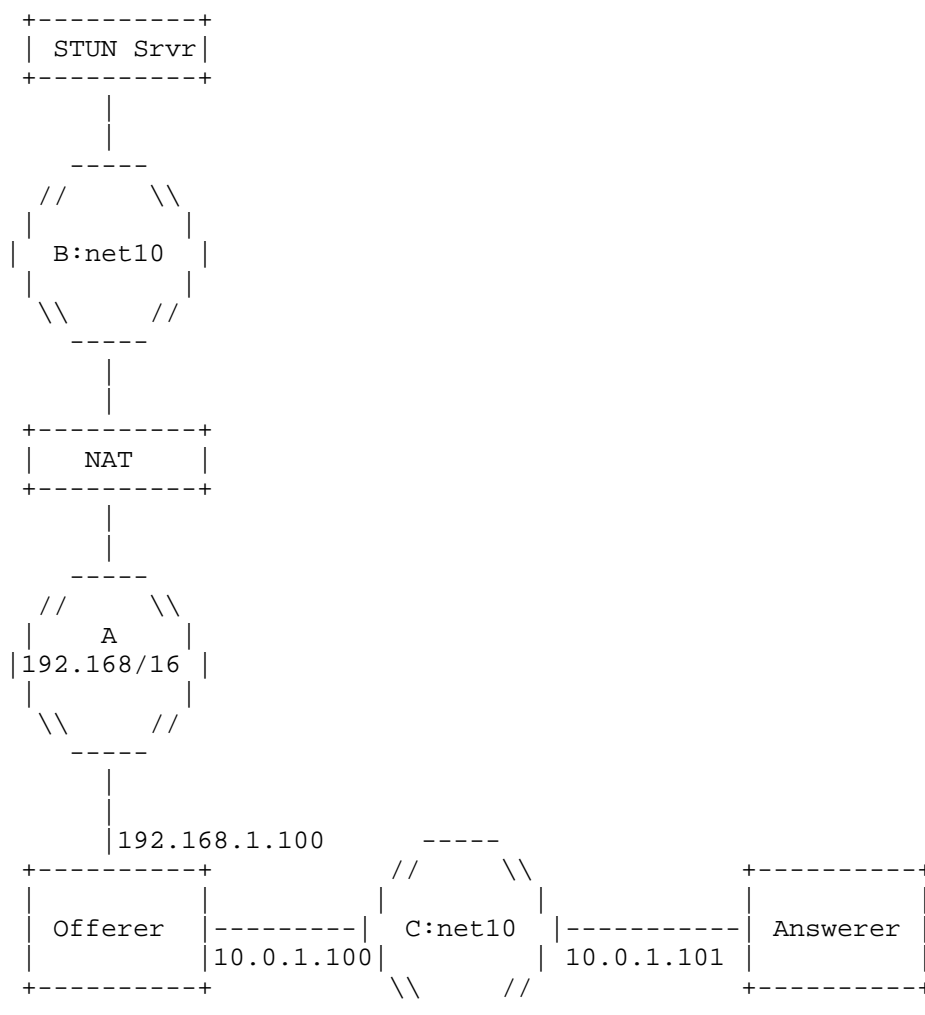


Figure 10: Identical Candidates with Different Bases

In this case, the offerer is multihomed. 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](mailto:jdrosen@jdrosen.net)  
URI: <http://www.jdrosen.net>





Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 28, 2013

S. Nandakumar  
C. Jennings  
Cisco  
February 24, 2013

A Framework for SDP Attributes when Multiplexing  
draft-nandakumar-mmusic-sdp-mux-attributes-01

Abstract

Communication sessions in RTCWeb are described and negotiated using Session Description Protocol(SDP)[RFC4566] mechanisms. SDP uses attributes to describes specific aspects of such a session. These attributes have different impacts when using varions of multiplexing RTP on a single transport layer flow. The scope of this specification is to provide a framework for analyzing the multiplex characteristics for SDP attributes and provide multiplexing characteristics of existing attributes.

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 28, 2013.

Copyright Notice

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

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	4
2. Terminology . . . . .	4
3. Motivation . . . . .	4
4. SDP Attribute Analysis Framework . . . . .	5
5. Analysis of Existing Attributes . . . . .	6
5.1. RFC4566 . . . . .	6
5.2. RFC4585 . . . . .	7
5.3. RFC5761 . . . . .	8
5.4. RFC4574 . . . . .	8
5.5. RFC5432 . . . . .	9
5.6. RFC4568 . . . . .	9
5.7. RFC5762 . . . . .	10
5.8. RFC6773 . . . . .	10
5.9. RFC5506 . . . . .	10
5.10. RFC6787 . . . . .	11
5.11. RFC5245 . . . . .	11
5.12. RFC5285 . . . . .	12
5.13. RFC3605 . . . . .	13
5.14. RFC5576 . . . . .	13
5.15. RFC6236 . . . . .	14
5.16. RFC6285 . . . . .	15
5.17. RFC6230 . . . . .	15
5.18. RFC6364 . . . . .	15
5.19. RFC4796 . . . . .	16
5.20. RFC3407 . . . . .	16
5.21. RFC6284 . . . . .	17
5.22. RFC6714 . . . . .	18
5.23. RFC4583 . . . . .	18
5.24. RFC5547 . . . . .	19
5.25. draft-ietf-mmusic-media-loopback . . . . .	19
5.26. RFC5760 . . . . .	20
5.27. RFC3611 . . . . .	20
5.28. RFC5939 . . . . .	21
5.29. draft-ietf-mmusic-sdp-media-capabilities . . . . .	21
5.30. RFC4567 . . . . .	22
5.31. RFC4572 . . . . .	23
5.32. RFC4570 . . . . .	23
5.33. RFC6128 . . . . .	24
5.34. RFC6189 . . . . .	24
5.35. RFC4145 . . . . .	24
5.36. RFC5159 . . . . .	25

5.37. RFC6193	26
5.38. RFC6064	26
5.39. RFC3108	29
5.40. 3GPP TS 24.182	32
5.41. 3GPP TS 24.183	32
5.42. 3GPP TS 24.229	32
5.43. ITU T.38	33
5.44. ITU-T H.248.15	34
5.45. RFC4975	34
5.46. Unknowns	35
6. bwtpe Attribute Analysis	36
6.1. RFC4566	36
6.2. RFC3556	36
6.3. RFC3890	36
7. rtcp-fb Attribute Analysis	37
7.1. RFC4585	37
7.2. RFC5104	37
8. rtcp-fb "ack/nack" Attribute Analysis	38
8.1. RFC4585	38
8.2. RFC6285	39
8.3. RFC6679	39
8.4. RFC6642	39
9. Codec Control Messages Analysis	40
9.1. RFC5104	40
10. group Attribute Analysis	40
10.1. RFC5888	40
10.2. RFC3524	41
10.3. RFC4091	41
10.4. RFC5956	42
10.5. RFC5583	42
11. ssrc-group Attribute Analysis	42
11.1. RFC5576	43
12. QoS Mechanism Token Analysis	43
12.1. RFC5432	43
13. k= Attribute Analysis	43
13.1. RFC4566	43
14. content Attribute Analysis	44
14.1. RFC4796	44
15. Payload Formats	44
15.1. RFC5109	44
16. TRANSPORT Category Example	45
17. IANA Considerations	46
18. Security Considerations	46
19. Change Log	46
20. References	46
20.1. Normative References	46
20.2. Informative References	46
Authors' Addresses	52

## 1. Introduction

SDP[RFC4566] defines several attributes for describing specific aspects of a multimedia session. Given the number of SDP attributes registered with IANA [IANA] and the defining new attributes in the future, there is need for future-proof framework to analyze these attributes for their applicability in transport multiplexing use-cases. Few proposed schemes for the same are defined in [I-D.ietf-mmusic-sdp-bundle-negotiation] and [I-D.ietf-avt-multiplexing-rtp]

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 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 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 180 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 multiplexing.

#### 4. SDP Attribute Analysis Framework

An SDP session description consists of a session-level section followed by zero or more media-level sections that could be broken down into following high-level categories

- o Attributes related to media content such as media type, encoding schemes, payload types to name a few.
- o Attributes specifying media transport characteristics like RTP/RTCP port, network addresses, QOS and so on.
- o Attributes Metadata descriptions such as session timing and origin information.
- o Attributes establishing relationships between media streams.

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 .

**BAD** Attributes where multiplexing SHOULD NOT be used if these attributes are in use in the SDP.

**IDENTICAL** Attributes that MUST be identical across all the media lines being multiplexed.

**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.

**TRANSPORT** Attributes that can be set normally for multiple items in a multiplexed group but the software MUST pick just one of the attribute of the given type for use. The one chosen is the attribute associated with the "m=" lines that represents the information being used for the transport of the RTP. Readers are advised to refer Section TRANSPORT\_EXAMPLE (Section 16) for an illustration of this category.

**SPECIAL** Attributes where the text in the draft must be consulted to see how they are handled when multiplexing.

The idea behind these categories is to provide recommendations for using the attributes when multiplexing is being used.

Section 5 analyzes attributes listed in IANA [IANA] grouped under the IETF document that defines them. The "Current" column indicates whether the attribute is currently specified as:

- o S -- Session level
- o M -- Media level
- o B -- Both
- o SR -- Source-level (for a single SSRC)

## 5. Analysis of Existing Attributes

### 5.1. RFC4566

RFC4566 [RFC4566] defines the Session Description Protocol (SDP) that is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation

Attr Name	Notes	Current	Category
sendrecv	Not impacted	B	NORMAL
sendonly	Not impacted	B	NORMAL
recvonly	Not impacted	B	NORMAL
inactive	Not impacted	B	NORMAL
cat	Not impacted	S	NORMAL
ptime	Not Impacted	M	NORMAL
maxptime	Not Impacted	M	NORMAL
orient	Not Impacted	M	NORMAL
framerate	Not Impacted	M	NORMAL
quality	Not Impacted	M	NORMAL
rtpmap	Not Impacted	M	NORMAL
fntp	Not Impacted	M	NORMAL
keywds	Not impacted	S	NORMAL
type	Not Impacted	S	NORMAL
tool	Not Impacted	S	NORMAL
charset	Not Impacted	S	NORMAL
sdplang	Not Impacted	B	NORMAL
lang	Not Impacted	B	NORMAL

## RFC4566 Attribute Analysis

## 5.2. RFC4585

RFC4585 [RFC4585] defines an extension to the Audio-visual Profile (AVP) that enables receivers to provide, statistically, more immediate feedback to the senders and thus allows for short-term

adaptation and efficient feedback-based repair mechanisms to be implemented.

Attr Name	Notes	Current	Category
rtcp-fb	Since RTCP feedback are reported per RTP Session, this attribute should be repeated across m= lines	M	IDENTICAL

#### RFC4585 Attribute Analysis

#### 5.3. RFC5761

RFC5761 [RFC5761] discusses issues that arise when multiplexing RTP data packets and RTP Control Protocol (RTCP) packets on a single UDP port. It describes when such multiplexing is and is not appropriate, and it explains how the Session Description Protocol (SDP) can be used to signal multiplexed sessions.

Name	Notes	Current	Category
rtcp-mux	RTCP muxing should be repeated across all the m=lines	M	IDENTICAL

#### RFC5761 Attribute Analysis

#### 5.4. RFC4574

RFC4574 [RFC4574] defines a new Session Description Protocol (SDP) media-level attribute: "label". The "label" attribute carries a pointer to a media stream in the context of an arbitrary network application that uses SDP. The sender of the SDP document can attach the "label" attribute to a particular media stream or streams. The application can then use the provided pointer to refer to each particular media stream in its context.



Name	Notes	Current	Category
label	Not Impacted	M	NORMAL

## RFC4574 Attribute Analysis

## 5.5. RFC5432

RFC5432 [RFC5432] defines prordures to negotiate QOS mechanisms using the Session Description Protocol (SDP) offer/answer model.

Name	Notes	Current	Category
qos-mech-send	QOS mechanism should be same across all the m=lines multiplexed	B	IDENTICAL
qos-mech-recv	AQOS mechanism should be same across all the m=lines multiplexed	B	IDENTICAL

## RFC5432 Attribute Analysis

## 5.6. RFC4568

RFC4568 [RFC4568] defines a Session Description Protocol (SDP) cryptographic attribute for unicast media streams. The attribute describes a cryptographic key and other parameters that serve to configure security for a unicast media stream in either a single message or a roundtrip exchange.

Name	Notes	Current	Category
crypto		M	TRANSPORT

## RFC4568 Attribute Analysis

Open Isuse: should this be NORMAL

## 5.7. RFC5762

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	Not recommended due to DCCP service code mismatch between different media types	M	BAD

## RFC5762 Attribute Analysis

## 5.8. RFC6773

RFC6773 [RFC6773] document specifies an alternative encapsulation of the Datagram Congestion Control Protocol (DCCP), referred to as DCCP-UDP. This encapsulation allows DCCP to be carried through the current generation of Network Address Translation (NAT) middleboxes without modification of those middleboxes

Name	Notes	Current	Category
dccp-port	Not recommended due to DCCP service code mismatch between different media types	M	BAD

## RFC6773 Attribute Analysis

## 5.9. RFC5506

RFC5506 [RFC5506] discusses benefits and issues that arise when allowing Real-time Transport Protocol (RTCP) packets to be transmitted with reduced size.

Name	Notes	Current	Category
rtcp-rsize	RTCP reduced size MUST be repeated across all the m=lines	M	IDENTICAL

## RFC5506 Attribute Analysis

## 5.10. RFC6787

The Media Resource Control Protocol Version 2 (MRCPv2) allows client hosts to control media service resources such as speech synthesizers, recognizers, verifiers, and identifiers residing in servers on the network. MRCPv2 is not a "stand-alone" protocol -- it relies on other protocols, such as the Session Initiation Protocol (SIP), to coordinate MRCPv2 clients and servers and manage sessions between them, and the Session Description Protocol (SDP) to describe, discover, and exchange capabilities. It also depends on SIP and SDP to establish the media sessions and associated parameters between the media source or sink and the media server. Once this is done, the MRCPv2 exchange operates over the control session established above, allowing the client to control the media processing resources on the speech resource server. RFC6787 [RFC6787] defines attributes for this purpose.

Name	Notes	Current	Category
resource	Not Impacted	M	NORMAL
channel	Not Impacted	M	NORMAL
	Not Impacted	M	NORMAL

## RFC6787 Attribute Analysis

## 5.11. RFC5245

RFC5245 [RFC5245] describes a protocol for Network Address Translator(NAT) traversal for UDP-based multimedia sessions established with the offer/answer model. This protocol is called Interactive Connectivity Establishment (ICE). ICE makes use of the Session Traversal Utilities for NAT (STUN) protocol and its

extension, Traversal Using Relay NAT (TURN). ICE can be used by any protocol utilizing the offer/answer model, such as the Session Initiation Protocol (SIP).

Name	Notes	Current	Category
ice-lite	Not Impacted	S	NORMAL
ice-options	Not Impacted	S	NORMAL
ice-options	Not Impacted	S	NORMAL
ice-pwd	Per media-level attribute MUST be used per underlying transport flow	B	TRANSPORT
ice-ufrag	Per media-level attribute MUST be used per underlying transport flow	B	TRANSPORT
candidate	Per media-level attribute MUST be used per underlying transport flow		TRANSPORT
remote-candidates	Per media-level attribute MUST be used per underlying transport flow	M	TRANSPORT

#### RFC5245 Attribute Analysis

#### 5.12. RFC5285

RFC5285 [RFC5285] provides a general mechanism to use the header extension feature of RTP (the Real-Time Transport Protocol). It provides the option to use a small number of small extensions in each RTP packet, where the universe of possible extensions is large and registration is de-centralized. The actual extensions in use in a session are signaled in the setup information for that session.

Name	Notes	Current	Category
extmap	Specific RTP extension document MUST be referred	B	SPECIAL

## RFC5285 Attribute Analysis

## 5.13. RFC3605

Originally, SDP assumed that RTP and RTCP were carried on consecutive ports. However, this is not always true when NATs are involved. [RFC3605] specifies an early mechanism to indicate the RTCP port.

Name	Notes	Current	Category
rtcp	Case1:Same RTCP port is repeated across the m=lines. Case2:Different RTCP ports renders multiplexing impossible	M	TRANSPORT

## RFC3605 Attribute Analysis

## 5.14. RFC5576

RFC5576 [RFC5576] defines a mechanism to describe RTP media sources, which are identified by their synchronization source (SSRC) identifiers, in SDP, to associate attributes with these sources, and to express relationships among sources. It also defines several source-level attributes that can be used to describe properties of media sources.

Name	Notes	Current	Category
ssrc	SSRCs repeated over multiple m=lines is forbidden if the m-lines are in the same RTP session.	M	BAD
ssrc-group	Refer to section Section 11 for specific analysis of the grouping semantics	M	SPECIAL
cname	Not Impacted [Open Issues: what are the rules for CNAME duplication across sessions?]	SR	NORMAL
previous-ssrc	SSRCs repeated over multiple m=lines complicates multiplexing	SR	BAD
fntp	Not Impacted	SR	NORMAL

## RFC5576 Attribute Analysis

## 5.15. RFC6236

RFC6236 [RFC6236] proposes a new generic session setup attribute to make it possible to negotiate different image attributes such as image size. A possible use case is to make it possible for a low-end hand-held terminal to display video without the need to rescale the image, something that may consume large amounts of memory and processing power. The document also helps to maintain an optimal bitrate for video as only the image size that is desired by the receiver is transmitted.

Name	Notes	Current	Category
imageattr	Not Impacted	M	NORMAL

## RFC6236 Attribute Analysis

## 5.16. RFC6285

RFC6285 [RFC6285] describes a method using the existing RTP and RTP Control Protocol (RTCP) machinery that reduces the acquisition delay. In this method, an auxiliary unicast RTP session carrying the Reference Information to the receiver precedes or accompanies the multicast stream. This unicast RTP flow can be transmitted at a faster than natural bitrate to further accelerate the acquisition. The motivating use case for this capability is multicast applications that carry real-time compressed audio and video.

Name	Notes	Current	Category
rams-updates	Not recommended	M	BAD

## RFC6285 Attribute Analysis

## 5.17. RFC6230

RFC6230 [RFC6230] describes a framework and protocol for application deployment where the application programming logic and media processing are distributed. This implies that application programming logic can seamlessly gain access to appropriate resources that are not co-located on the same physical network entity. The framework uses the Session Initiation Protocol (SIP) to establish an application-level control mechanism between application servers and associated external servers such as media servers.

Name	Notes	Current	Category
cfw-id	Not Applicable	M	NORMAL

## RFC6230 Attribute Analysis

## 5.18. RFC6364

RFC6364 [RFC6364] specifies the use of the Session Description Protocol (SDP) to describe the parameters required to signal the Forward Error Correction (FEC) Framework Configuration Information between the sender(s) and receiver(s). This document also provides examples that show the semantics for grouping multiple source and repair flows together for the applications that simultaneously use

multiple instances of the FEC Framework.

Name	Notes	Current	Category
fec-source-flow	Not Impacted	M	NORMAL
fec-repair-flow	Not Impacted	M	NORMAL
repair-window	Not Impacted	M	NORMAL

#### RFC6364 Attribute Analysis

#### 5.19. RFC4796

RFC4796 [RFC4796] defines a new Session Description Protocol (SDP) media- level attribute, 'content'. The 'content' attribute defines the content of the media stream to a more detailed level than the media description line. The sender of an SDP session description can attach the 'content' attribute to one or more media streams. The receiving application can then treat each media stream differently (e.g., show it on a big or small screen) based on its content.

Name	Notes	Current	Category
content	Not Impacted	M	NORMAL

#### RFC4796 Attribute Analysis

#### 5.20. RFC3407

RFC3407 [RFC3407] defines a set of Session Description Protocol (SDP) attributes that enables SDP to provide a minimal and backwards compatible capability declaration mechanism.



Name	Notes	Current	Category
sqn	Not Impacted	B	NORMAL
csdc	Mismatch in the offered capability description MAY fail multiplexing.	B	BAD
cpar	Mismatch in the offered capability parameters MAY fail multiplexing.	B	BAD
cparmin	Mismatch in the offered capability parameters MAY fail multiplexing.	B	BAD
cparmax	Mismatch in the offered capability parameters MAY fail multiplexing.	B	BAD

## RFC3407 Attribute Analysis

## 5.21. RFC6284

RFC6284 [RFC6284] presents a port mapping solution that allows RTP receivers to choose their own ports for an auxiliary unicast session in RTP applications using both unicast and multicast services. The solution provides protection against denial-of-service or packet amplification attacks that could be used to cause one or more RTP packets to be sent to a victim client

Name	Notes	Current	Category
portmapping-req	Not recommended, if port mapping is required by the application	M	BAD

## RFC6284 Attribute Analysis

## 5.22. RFC6714

RFC6714 [RFC6714] defines a Message Session Relay Protocol (MSRP) extension, Connection Establishment for Media Anchoring (CEMA). Support of this extension is OPTIONAL. The extension allows middleboxes to anchor the MSRP connection, without the need for middleboxes to modify the MSRP messages; thus, it also enables secure end-to-end MSRP communication in networks where such middleboxes are deployed. This document also defines a Session Description Protocol (SDP) attribute, 'msrp-cema', that MSRP endpoints use to indicate support of the CEMA extension.

Name	Notes	Current	Category
msrp-cema	Not recommended due to legacy interop purposes	M	NORMAL

## RFC6714 Attribute Analysis

## 5.23. RFC4583

RFC4583 [RFC4583] document specifies how to describe Binary Floor Control Protocol (BFCP) streams in Session Description Protocol (SDP) descriptions. User agents using the offer/answer model to establish BFCP streams use this format in their offers and answers

Name	Notes	Current	Category
floorctrl	Not Impacted	M	NORMAL
confid	Not Impacted	M	NORMAL
userid	Not Impacted	M	NORMAL
floorid	Not Impacted	M	NORMAL

## RFC4583 Attribute Analysis

## 5.24. RFC5547

RFC5547 [RFC5547] provides a mechanism to negotiate the transfer of one or more files between two endpoints by using the Session Description Protocol (SDP) offer/answer model specified in [RFC3264].

Name	Notes	Current	Category
file-selector	Not Impacted	M	NORMAL
file-transfer-id	Not Impacted	M	NORMAL
file-disposition	Not Impacted	M	NORMAL
file-date,file-iconfile-range	Not Impacted	M	NORMAL
file-iconfile-range	Not Impacted	M	NORMAL
file-iconfile-range	Not Impacted	M	NORMAL

## RFC5547 Attribute Analysis

## 5.25. draft-ietf-mmusic-media-loopback

[MEDIA\_LOOPBACK] adds new SDP media types and attributes, which enable establishment of media sessions where the media is looped back to the transmitter. Such media sessions will serve as monitoring and troubleshooting tools by providing the means for measurement of more advanced VoIP, Real-time Text and Video over IP performance metrics.

Name	Notes	Current	Category
loopback rtp-pkt-loopback	The attribute MUST be repeated across all m=lines multiplexed	M	IDENTICAL
loopback rtp-media-loopback	Not Impacted	M	NORMAL
loopback-source	Not Impacted	M	NORMAL
loopback-mirror	Not Impacted	M	NORMAL

## draft-ietf-mmusic-media-loopback Attribute Analysis

## 5.26. RFC5760

RFC5760 [RFC5760] specifies an extension to the Real-time Transport Control Protocol (RTCP) to use unicast feedback to a multicast sender. The proposed extension is useful for single-source multicast sessions such as Source-Specific Multicast (SSM) communication where the traditional model of many-to-many group communication is either not available or not desired.

Name	Notes	Current	Category
rtcp-unicast	The attribute MUST be reported across all m=lines multiplexed	M	IDENTICAL

## RFC5760 Attribute Analysis

## 5.27. RFC3611

RFC3611 [RFC3611] defines the Extended Report (XR) packet type for the RTP Control Protocol (RTCP), and defines how the use of XR packets can be signaled by an application if it employs the Session Description Protocol (SDP).

Name	Notes	Current	Category
rtcp-xr	The attribute MUST be reported across all m=lines multiplexed	B	IDENTICAL

## RFC3611 Attribute Analysis

## 5.28. RFC5939

RFC5939 [RFC5939] defines a general SDP Capability Negotiation framework. It also specifies how to provide attributes and transport protocols as capabilities and negotiate them using the framework. Extensions for other types of capabilities (e.g., media types and media formats) may be provided in other documents.

Name	Notes	Current	Category
pcfg	Depends on capability being negotiated	M	SPECIAL
acfg	Depends on capability being negotiated	M	SPECIAL
csup	Depends on capability being negotiated	B	SPECIAL
creq	Depends on capability being negotiated	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	Current	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

RFC4567 [RFC4567] defines general extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP) to carry messages, as specified by a key management protocol, in order to secure the media. These extensions are presented as a framework, to be used by one or more key management protocols. As such, their use is meaningful only when complemented by an appropriate key management protocol.

Name	Notes	Current	Category
key-mgmt	Key management protocol MUST be identical across all the m=lines	B	IDENTICAL

#### RFC4567 Attribute Analysis

## 5.31. RFC4572

RFC4572 [RFC4572] specifies how to establish secure connection-oriented media transport sessions over the Transport Layer Security (TLS) protocol using the Session Description Protocol (SDP). It defines a new SDP protocol identifier, 'TCP/TLS'. It also defines the syntax and semantics for an SDP 'fingerprint' attribute that identifies the certificate that will be presented for the TLS session. This mechanism allows media transport over TLS connections to be established securely, so long as the integrity of session descriptions is assured.

Name	Notes	Current	Category
fingerprint	Fingerprint value MUST be identical across all the m=lines	B	IDENTICAL

## RFC4572 Attribute Analysis

## 5.32. RFC4570

RFC4570 [RFC4570] describes how to adapt the Session Description Protocol (SDP) to express one or more source addresses as a source filter for one or more destination "connection" addresses. It defines the syntax and semantics for an SDP "source-filter" attribute that may reference either IPv4 or IPv6 address(es) as either an inclusive or exclusive source list for either multicast or unicast destinations. In particular, an inclusive source-filter can be used to specify a Source-Specific Multicast (SSM) session

Name	Notes	Current	Category
source-filter	he attribute MUST be repeated across all m=lines multiplexed	B	IDENTICAL

## RFC4570 Attribute Analysis

## 5.33. RFC6128

The Session Description Protocol (SDP) has an attribute that allows RTP applications to specify an address and a port associated with the RTP Control Protocol (RTCP) traffic. In RTP-based source-specific multicast (SSM) sessions, the same attribute is used to designate the address and the RTCP port of the Feedback Target in the SDP description. However, the RTCP port associated with the SSM session itself cannot be specified by the same attribute to avoid ambiguity, and thus, is required to be derived from the "m=" line of the media description. Deriving the RTCP port from the "m=" line imposes an unnecessary restriction. RFC6128 [RFC6128] removes this restriction by introducing a new SDP attribute.

Name	Notes	Current	Category
multicast-rtcp	Multicast RTCP port MUST be identical across all the m=lines	B	IDENTICAL

## RFC6128 Attribute Analysis

## 5.34. RFC6189

RFC6189 [RFC6189] defines ZRTP, a protocol for media path Diffie-Hellman exchange to agree on a session key and parameters for establishing unicast Secure Real-time Transport Protocol (SRTP) sessions for Voice over IP (VoIP) applications.

Name	Notes	Current	Category
zrtp-hash	Complicates if all the m=lines are not authenticated	M	BAD

## RFC6189 Attribute Analysis

## 5.35. RFC4145

RFC4145 [RFC4145] describes how to express media transport over TCP using the Session Description Protocol (SDP). It defines the SDP 'TCP' protocol identifier, the SDP 'setup' attribute, which describes the connection setup procedure, and the SDP 'connection' attribute,



which handles connection reestablishment.

Name	Notes	Current	Category
setup	Not recommended for multiplexing due to possible differences in the protocol and media types	B	BAD
connection	Not recommended for multiplexing due to possible differences in the protocol and media types	B	BAD

#### RFC4145 Attribute Analysis

##### 5.36. RFC5159

RFC5159 [RFC5159] provides descriptions of Session Description Protocol (SDP) attributes used by the Open Mobile Alliance's Broadcast Service and Content Protection specification.

Name	Notes	Current	Category
bcastversion	Not recommended for multiplexing for legacy interop purposes	S	BAD
stkmstream	Not recommended for multiplexing for legacy interop purposes	B	BAD
SRTPAuthentication	Not recommended for multiplexing for legacy interop purposes	M	BAD
SRTPROCTxRate	Not recommended for multiplexing for legacy interop purposes	M	BAD

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

## RFC5159 Attribute Analysis

## 5.37. RFC6193

RFC6193 [RFC6193] specifies how to establish a media session that represents a virtual private network using the Session Initiation Protocol for the purpose of on-demand media/application sharing between peers. It extends the protocol identifier of the Session Description Protocol (SDP) so that it can negotiate use of the Internet Key Exchange Protocol (IKE) for media sessions in the SDP offer/answer model.

Name	Notes	Current	Category
ike-setup	Attribute MUST be identical across all the m=lines	B	IDENTICAL
psk-fingerprint	Attribute MUST be identical across all the m=lines	B	IDENTICAL
ike-esp	Attribute MUST be identical across all the m=lines	B	IDENTICAL
ike-esp-udpencap	Attribute MUST be identical across all the m=lines	B	IDENTICAL

## RFC6193 Attribute Analysis

In the case of session multiplexed with multiple m=lines, this SHOULD create only one IPSEC association for all the m=lines.

## 5.38. RFC6064

The Packet-switched Streaming Service (PSS) and the Multimedia Broadcast/Multicast Service (MBMS) defined by 3GPP use the Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP) with some extensions. RFC6064 [RFC6064] provides information about these extensions and registers the RTSP and SDP extensions with IANA.

Name	Notes	Current	Category
X-predecbufsize	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing with audio m=lines is invalid	M	BAD
X-initpredecbufperiod	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing with audio m=lines is invalid	M	BAD
X-initpostdecbufperiod	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing with audio m=lines is invalid	M	BAD
X-decbyterate	Case1:Aggregate total when video m-lines are muxed Case2:Multiple xing with audio m=lines is invalid	M	BAD
3gpp-videopostdecbufsize	Case1:Aggregate total when video m-lines are muxed. Case2:Multiplexing with audio m=lines is invalid	M	BAD
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	BAD
alt-group	Complicates selection of alternate m=lines grouped with alt-group on mulitplexing	M	BAD
3GPP-Adaptation-Support	Not recommended for legacy interop purposes	M	BAD
3GPP-QoE-Metricsn	Not recommended for legacy interop purposes	B	BAD
3GPP-Asset-Informatio	Not recommended for legacy interop purposes	B	BAD
mbms-mode	Not recommended for legacy interop purposes	B	BAD
mbms-flowid	Multiplexing multiple m=lines complicates FEC mappings to the transport addresses.	M	BAD
mbms-repair	Not recommended for legacy interop purposes	B	BAD

## RFC6064 Attribute Analysis

## 5.39. RFC3108

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	Current	Category
aalType	Not recommended for legacy interop purposes	M	BAD
eecid	Not recommended for legacy interop purposes	M	BAD
aalType	Not recommended for legacy interop purposes	M	BAD
capability	Not recommended for legacy interop purposes	M	BAD
qosClass	Not recommended for legacy interop purposes	M	BAD
bcob	Not recommended for legacy interop purposes	M	BAD
stc	Not recommended for legacy interop purposes	M	BAD
upcc	Not recommended for legacy interop purposes	M	BAD
atmQOSparms	Not recommended for legacy interop purposes	M	BAD
atmTrfcDesc	Not recommended for legacy interop purposes	M	BAD
abrParms	Not recommended for legacy interop purposes	M	BAD
abrSetup	Not recommended for legacy interop purposes	M	BAD
bearerType	Not recommended for legacy interop purposes	M	BAD

lij	Not recommended for legacy interop purposes	M	BAD
anycast	Not recommended for legacy interop purposes	M	BAD
cache	Not recommended for legacy interop purposes	M	BAD
bearerSigIE	Not recommended for legacy interop purposes	M	BAD
aalApp	Not recommended for legacy interop purposes	M	BAD
cbrRate	Not recommended for legacy interop purposes	M	BAD
sbc	Not recommended for legacy interop purposes	M	BAD
clkrec	Not recommended for legacy interop purposes	M	BAD
fec	Not recommended for legacy interop purposes	M	BAD
prtfl	Not recommended for legacy interop purposes	M	BAD
structure	Not recommended for legacy interop purposes	M	BAD
cpsSDUsize	Not recommended for legacy interop purposes	M	BAD
aal2CPS	Not recommended for legacy interop purposes	M	BAD
aal2CPSSDURate	Not recommended for legacy interop purposes	M	BAD
aal2sscs3661unassured	Not recommended for legacy interop purposes	M	BAD
aal2sscs3661assured	Not recommended for legacy interop purposes	M	BAD

aal2sscs3662	Not recommended for legacy interop purposes	M	BAD
aal5sscop	Not recommended for legacy interop purposes	M	BAD
atmmap	Not recommended for legacy interop purposes	M	BAD
silenceSupp	Not recommended for legacy interop purposes	M	BAD
ecan	Not recommended for legacy interop purposes	M	BAD
gc	Not recommended for legacy interop purposes	M	BAD
profileDesc	Not recommended for legacy interop purposes	M	BAD
vsel	Not recommended for legacy interop purposes	M	BAD
dsel	Not recommended for legacy interop purposes	M	BAD
fsel	Not recommended for legacy interop purposes	M	BAD
onewaySel	Not recommended for legacy interop purposes	M	BAD
codeccconfig	Not recommended for legacy interop purposes	M	BAD
isup_usi	Not recommended for legacy interop purposes	M	BAD
isup_usi	Not recommended for legacy interop purposes	M	BAD
chain	Not recommended for legacy interop purposes	M	BAD

## RFC3108 Attribute Analysis

## 5.40. 3GPP TS 24.182

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

Name	Notes	Current	Category
g.3gpp.cat	Not recommended due to interop purposes	M	BAD

## 3GPP TS 24.182 Attribute Analysis

## 5.41. 3GPP TS 24.183

3GPP TS 24.183 [3GPP TS 24.183] specifies IP multimedia subsystem Custom Ringing Signal

Name	Notes	Current	Category
g.3gpp.crs	Not recommended due to interop purposes	M	BAD

## 3GPP TS 24.183 Attribute Analysis

## 5.42. 3GPP TS 24.229

3GPP TS 24.229 [3GPP TS 24.229] IP multimedia call control protocol based on Session Initial protocol and Session Description Protocol.



Name	Notes	Current	Category
secondary-realm	Not recommended due to interop purposes	M	BAD
visited realm	Not recommended due to interop purposes	M	BAD
omr-m-cksum	Not recommended due to interop purposes	M	BAD
omr-s-cksum	Not recommended due to interop purposes	M	BAD
omr-m-att	Not recommended due to interop purposes	M	BAD
omr-s-att	Not recommended due to interop purposes	M	BAD
omr-s-bw	Not recommended due to interop purposes	M	BAD
omr-m-att	Not recommended due to interop purposes	M	BAD
omr-codecs	Not recommended due to interop purposes	M	BAD

## 3GPP TS 24.229 Attribute Analysis

## 5.43. ITU T.38

ITU T.38[T.38] defines procedures for real-time Group 3 facsimile communications over IP networks.

Name	Notes	Current	Category
T38FaxVersion	Not Impacted	S	NORMAL
T38MaxBitRate	Not Impacted	S	NORMAL
T38FaxFillBitRemoval	Not Impacted	S	NORMAL
T38FaxTranscodingMMR	Not Impacted	S	NORMAL
T38FaxTranscodingJBIG	Not Impacted	S	NORMAL
T38FaxRateManagement	Not Impacted	S	NORMAL
T38FaxMaxBuffer	Not Impacted	S	NORMAL
T38FaxMaxDatagram	Not Impacted	S	NORMAL
T38FaxUdpEC	Not Impacted	S	NORMAL

#### Historic Attribute Analysis

##### 5.44. ITU-T H.248.15

ITU-T H.248.15 [H.248.15] defines Gateway Control Protocol SDP H.248 package attribute

Name	Notes	Current	Category
h248item	Not recommended for interop purposes	S	BAD

#### Historic Attribute Analysis

##### 5.45. RFC4975

RFC4975 [RFC4975] the Message Session Relay Protocol, a protocol for transmitting a series of related instant messages in the context of a session. Message sessions are treated like any other media stream when set up via a rendezvous or session creation protocol such as the Session Initiation Protocol.

Name	Notes	Current	Category
accept-types	Not recommended due to incompatible media types	M	BAD
accept-wrapped-types	Not recommended due to incompatible media typess	M	BAD
max-size	Not recommended due to incompatible media types	M	BAD
path	Not recommended due to incompatible media types	M	BAD

## RFC4975 Attribute Analysis

## 5.46. Unknowns

This section specifies analysis for the attributes that are included for historic usage alone by the [IANA\_REF]

Name	Notes	Current	Category
rtpred1	Not Applicable	Not-Applicable	BAD
rtpred2	Not Applicable	Not-Applicable	BAD
PSCid	Not Applicable	Not-Applicable	BAD
bc_service	Not Applicable	Not-Applicable	BAD
bc_program	Not Applicable	Not-Applicable	BAD
bc_service_package	Not Applicable	Not-Applicable	BAD

## Unknowns Attribute Analysis

## 6. bwtype Attribute Analysis

This section specifies handling of specific bandwidth attributes when used in multiplexing scenarios.

### 6.1. RFC4566

Name	Notes	Current	Category
bwtype:CT	Aggregate bandwidth for the conference	S	NORMAL
bwtype:AS	Aggregate RTP session bandwidth	B	NORMAL,SUM

RFC4566 bwtype Analysis

### 6.2. RFC3556

RFC3556 [RFC3556] defines an extension to the Session Description Protocol (SDP) to specify two additional modifiers for the bandwidth attribute. These modifiers may be used to specify the bandwidth allowed for RTP Control Protocol (RTCP) packets in a Real-time Transport Protocol (RTP) session

Name	Notes	Current	Category
bwtype:RS		B	NORMAL,SUM;
bwtype:RR		B	NORMAL,SUM

RFC3556 bwtype Analysis

### 6.3. RFC3890

RFC3890 [RFC3890] defines a Session Description Protocol (SDP) Transport Independent Application Specific Maximum (TIAS) bandwidth modifier that does not include transport overhead; instead an additional packet rate attribute is defined. The transport independent bit-rate value together with the maximum packet rate can then be used to calculate the real bit-rate over the transport actually used.

Name	Notes	Current	Category
bwtype:TIAS	Application MUST SUM bandwidth from all m=lines of same type and MUST NOT use default values	B	SUM

## RFC3890 bwtype Analysis

## 7. rtcp-fb Attribute Analysis

This section analyzes rtcp-fb SDP attributes [RTCP-FB].

## 7.1. RFC4585

RFC4585 [RFC4585] defines an extension to the Audio-visual Profile (AVP) that enables receivers to provide, statistically, more immediate feedback to the senders and thus allows for short-term adaptation and efficient feedback-based repair mechanisms to be implemented.

Attr Name	Notes	Current	Category
ack	Not Impacted	M	NORMAL
app	Not Impacted	M	NORMAL
nack	Not Impacted	M	NORMAL
trr-int	Not Impacted	M	NORMAL

## RFC4585 Attribute Analysis

## 7.2. RFC5104

RFC5104 [RFC5104] specifies a few extensions to the messages defined in the Audio-Visual Profile with Feedback (AVPF). They are helpful primarily in conversational multimedia scenarios where centralized multipoint functionalities are in use. However, some are also usable in smaller multicast environments and point-to-point calls.

Attr Name	Notes	Current	Category
ccm	Not Impacted	M	Normal

## RFC5104 Attribute Analysis

## 8. rtcp-fb "ack/nack" Attribute Analysis

This section analyzes rtcp-fb SDP attributes specific to ack and nack feedback types [ACK-NACK].

## 8.1. RFC4585

RFC4585 [RFC4585] defines an extension to the Audio-visual Profile (AVP) that enables receivers to provide, statistically, more immediate feedback to the senders and thus allows for short-term adaptation and efficient feedback-based repair mechanisms to be implemented.

Attr Name	Notes	Current	Category
nack sli	Not Impacted	M	NORMAL
nack pli	Not Impacted	M	NORMAL
ack rpsi	Not Impacted	M	NORMAL
ack app	Feedback parameters MUST be handled in the app specific way when multiplexed	M	SPECIAL
nack rpsi	Not Impacted	M	NORMAL
nack app	Feedback parameters MUST be handled in the app specific way when multiplexed	M	SPECIAL

## RFC4585 Attribute Analysis

## 8.2. RFC6285

Name	Notes	Current	Category
nack rai	Not Impacted	M	NORMAL

## RFC6285 Attribute Analysis

## 8.3. RFC6679

RFC6679 [RFC6679] specifies how Explicit Congestion Notification (ECN) can be used with the Real-time Transport Protocol (RTP) running over UDP, using the RTP Control Protocol (RTCP) as a feedback mechanism. It defines a new RTCP Extended Report (XR) block for periodic ECN feedback, a new RTCP transport feedback message for timely reporting of congestion events, and a Session Traversal Utilities for NAT (STUN) extension used in the optional initialisation method using Interactive Connectivity Establishment (ICE)

Name	Notes	Current	Category
nack ecn	Complicates ECN marking when m=lines of different types are used	M	SPECIAL

## RFC6679 Attribute Analysis

## 8.4. RFC6642

In a large RTP session using the RTP Control Protocol (RTCP) feedback mechanism defined in RFC 4585 [RFC4585], a feedback target may experience transient overload if some event causes a large number of receivers to send feedback at once. This overload is usually avoided by ensuring that feedback reports are forwarded to all receivers, allowing them to avoid sending duplicate feedback reports. However, there are cases where it is not recommended to forward feedback reports, and this may allow feedback implosion. RFC6642 [RFC6642] memo discusses these cases and defines a new RTCP Third-Party Loss Report that can be used to inform receivers that the feedback target is aware of some loss event, allowing them to suppress feedback.

Associated Session Description Protocol (SDP) signaling is also defined.

Name	Notes	Current	Category
tllei	Not Impacted	M	NORMAL
pslei	Not Impacted	M	NORMAL

#### RFC6642 Attribute Analysis

### 9. Codec Control Messages Analysis

This section analyzes rtcp-fb Codec Control Message [CCM].

#### 9.1. RFC5104

Attr Name	Notes	Current	Category
fir	Not Impacted	M	NORMAL
tmmbr	Not Impacted	M	NORMAL
tstr	Not Impacted	M	NORMAL
vbcm	Not Impacted	M	NORMAL

#### RFC5104 Attribute Analysis

### 10. group Attribute Analysis

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

#### 10.1. RFC5888

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



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

## RFC5888 Attribute Analysis

## 10.2. RFC3524

RFC3524 [RFC3524] defines an extension to the Session Description Protocol (SDP) grouping framework. It allows requesting a group of media streams to be mapped into a single resource reservation flow. The SDP syntax needed is defined, as well as a new "semantics" attribute called Single Reservation Flow (SRF).

Name	Notes	Current	Category
group:SRF	Not Impacted	S	NORMAL

## RFC3524 Attribute Analysis

## 10.3. RFC4091

RFC4091 [RFC4091] defines the Alternative Network Address Types (ANAT) semantics for the Session Description Protocol (SDP) grouping framework. The ANAT semantics allow alternative types of network addresses to establish a particular media stream.

Name	Notes	Current	Category
group:ANAT	Not Impacted	S	BAD

## RFC4091 Attribute Analysis

## 10.4. RFC5956

RFC5956 [RFC5956] defines the semantics for grouping the associated source and FEC-based (Forward Error Correction) repair flows in the Session Description Protocol (SDP). The semantics defined in the document are to be used with the SDP Grouping Framework (RFC 5888). These semantics allow the description of grouping relationships between the source and repair flows when one or more source and/or repair flows are associated in the same group, and they provide support for additive repair flows. SSRC-level (Synchronization Source) grouping semantics are also defined in this document for Real-time Transport Protocol (RTP) streams using SSRC multiplexing.

Name	Notes	Current	Category
group:FEC-FR	Not Impacted	S	NORMAL

## RFC5956 Attribute Analysis

## 10.5. RFC5583

RFC5583 [RFC5583] defines semantics that allow for signaling the decoding dependency of different media descriptions with the same media type in the Session Description Protocol (SDP). This is required, for example, if media data is separated and transported in different network streams as a result of the use of a layered or multiple descriptive media coding process.

Name	Notes	Current	Category
depend lay	Not Impacted	M	NORMAL
depend mdc	Not Impacted	M	NORMAL

## RFC5583 Attribute Analysis

## 11. ssrc-group Attribute Analysis

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

## 11.1. RFC5576

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

RFC5576 Attribute Analysis

## 12. QoS Mechanism Token Analysis

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

## 12.1. RFC5432

Name	Notes	Current	Category
rsvp	Not Impacted	B	NORMAL
nsis	Not Impacted	B	NORMAL

RFC5432 Attribute Analysis

## 13. k= Attribute Analysis

## 13.1. RFC4566

Name	Notes	Current	Category
k=	It is NOT recommended to use this attribute	S	BAD

RFC4566 Attribute Analysis

## 14. content Attribute Analysis

## 14.1. RFC4796

Name	Notes	Current	Category
content:slides	Not Impacted	M	NORMAL
content:speaker	Not Impacted	M	NORMAL
content:main	Not Impacted	M	NORMAL
content:sl	Not Impacted	M	NORMAL
content:alt	Not Impacted	M	NORMAL

RFC4796 Attribute Analysis

## 15. Payload Formats

## 15.1. RFC5109

RFC4145 [RFC5109] describes a payload format for generic Forward Error Correction (FEC) for media data encapsulated in RTP. It is based on the exclusive-or (parity) operation. The payload format allows end systems to apply protection using various protection lengths and levels, in addition to using various protection group sizes to adapt to different media and channel characteristics. It enables complete recovery of the protected packets or partial recovery of the critical parts of the payload depending on the packet loss situation.

Name	Notes	Current	Category
audio/ulpfec	Not recommended for multiplexing due to reuse of SSRCs	M	BAD
video/ulpfec	Not recommended for multiplexing due to reuse of SSRCs	M	BAD

text/ulpfec	Not recommended for multiplexing due to reuse of SSRCS	M	BAD
application/ulpfec	Not recommended for multiplexing due to reuse of SSRCS	M	BAD

## RFC5109 Payload Format Analysis

Draft draft-lennox-payload-ulp-ssrc-mux proposes a simple fix to make it possible to use ULP with multiplexing and ULP is allowed when used with that.

## 16. TRANSPORT Category Example

The example below explains the usage of "TRANSPORT" category for RFC4586 "crypto" parameter when SDP Port number based multiplexing is performed for audio and video streams.

```
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 second,first
m=audio 49172 RTP/AVP 99
a=mid:one
a=crypto:1 AES_CM_128_HMAC_SHA1_80
    inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAWJSoj|2^20|1:32
a=rtpmap:99 iLBC/8000
m=video 51374 RTP/AVP 31
a=mid:two
a=crypto:1 AES_CM_128_HMAC_SHA1_80
    inline:EcGZiNWpFJhQXdspcllekcmVCNWpVLcfHAWJSoj|2^20|1:32
a=rtpmap:96 H261/90000
```

In this example, "a=crypto" attribute is defined for both the audio and the video m=lines. The one that MUST be used for the multiplexed RTP Session is the one that corresponds to m=line with mid "two" even though the audio m=line with mid "one" occurs ahead of it. This is due to BUNDLE grouping semantics [I-D.ietf-mmusic-sdp-bundle-negotiation] which mandates the values

from mid occurring first on the a=group:BUNDLE line to be considered for setting up the RTP Transport.

## 17. IANA Considerations

TBD

Future versions of this specification will ask the IANA to expand to attribute tables to include an extra column specifying categories from this draft.

## 18. Security Considerations

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

## 19. Change Log

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

Changes from draft-nandakumar-mmusic-mux-attributes-00

- o Added new section for dealing with FEC payload types.

## 20. References

### 20.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

### 20.2. Informative References

[3GPP TS 24.182]  
"IP Multimedia Subsystem (IMS) Customized Alerting Tones (CAT); Protocol specification",  
<<http://www.3gpp.org/ftp/Specs/html-info/24182.htm>>.

[3GPP TS 24.183]  
"IP Multimedia Subsystem (IMS) Customized Ringing Signal (CRS); Protocol specification",  
<<http://www.3gpp.org/ftp/Specs/html-info/24183.htm>>.

- [3GPP TS 24.229]  
"IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP);",  
<<http://www.3gpp.org/ftp/Specs/html-info/24229.htm>>.
- [ACK-NACK]  
"Session Description Protocol (SDP) RTCP ACK/NACK Feedback attributes", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-15>>.
- [CCM]  
"Session Description Protocol (SDP) RTCP-FB Codec Control Messages", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-19>>.
- [GROUP-SEM]  
"Session Description Protocol (SDP) "group" semantics", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-13>>.
- [H.248.15]  
"Gateway control protocol: SDP H.248 package attribute",  
<<http://www.itu.int/rec/T-REC-H.248.15>>.
- [I-D.ietf-avt-multiplexing-rtp]  
El-Khatib, K., Luo, G., Bochmann, G., and Pinjiang. Feng,  
"Multiplexing Scheme for RTP Flows between Access Routers", Internet-Draft <http://tools.ietf.org/html/draft-ietf-avt-multiplexing-rtp-01>, October 1999.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]  
Holmberg, C., Alvestrand, H., and C. Jennings,  
"Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers",  
draft-ietf-mmusic-sdp-bundle-negotiation-03 (work in progress), February 2013.
- [IANA]  
"Session Description Protocol (SDP) Parameters", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml>>.
- [MEDIA\_CAP]  
Kaplan, H., Hedayat, K., and N. Venna, "Session Description Protocol (SDP) Media Capabilities Negotiation", draft-ietf-mmusic-sdp-media-capabilities-17 (work in progress), January 2013.
- [MEDIA\_LOOPBACK]

Kaplan, H., Hedayat, K., Venna, N., Jones, P., and N. Stratton, "An Extension to the Session Description Protocol (SDP) and Real-time Transport Protocol (RTP) for Media Loopback", draft-ietf-mmusic-media-loopback-27 (work in progress), January 2013.

[QOS] "Session Description Protocol (SDP) QoS Mechanism Tokens", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-20>>.

[RFC3108] Kumar, R. and M. Mostafa, "Conventions for the use of the Session Description Protocol (SDP) for ATM Bearer Connections", RFC 3108, May 2001.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3407] Andreassen, F., "Session Description Protocol (SDP) Simple Capability Declaration", RFC 3407, October 2002.

[RFC3524] Camarillo, G. and A. Monrad, "Mapping of Media Streams to Resource Reservation Flows", RFC 3524, April 2003.

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.

[RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[RFC3890] Westerlund, M., "A Transport Independent Bandwidth Modifier for the Session Description Protocol (SDP)", RFC 3890, September 2004.

[RFC4091] Camarillo, G. and J. Rosenberg, "The Alternative Network Address Types (ANAT) Semantics for the Session Description Protocol (SDP) Grouping Framework", RFC 4091, June 2005.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.



- [RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.
- [RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4570] Quinn, B. and R. Finlayson, "Session Description Protocol (SDP) Source Filters", RFC 4570, July 2006.
- [RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, July 2006.
- [RFC4574] Levin, O. and G. Camarillo, "The Session Description Protocol (SDP) Label Attribute", RFC 4574, August 2006.
- [RFC4583] Camarillo, G., "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", RFC 4583, November 2006.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.
- [RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5159] Dondeti, L. and A. Jerichow, "Session Description Protocol (SDP) Attributes for Open Mobile Alliance (OMA) Broadcast (BCAST) Service and Content Protection", RFC 5159, March 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment

(ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, July 2006.

- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5432] Polk, J., Dhesikan, S., and G. Camarillo, "Quality of Service (QoS) Mechanism Selection in the Session Description Protocol (SDP)", RFC 5432, March 2009.
- [RFC5506] Johansson, I., "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5762] Perkins, C., "RTP and the Datagram Congestion Control Protocol (DCCP)", RFC 5762, April 2010.
- [RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, September 2010.

- [RFC5956]   Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.
- [RFC6064]   Westerlund, M. and P. Frojdh, "SDP and RTSP Extensions Defined for 3GPP Packet-Switched Streaming Service and Multimedia Broadcast/Multicast Service", RFC 6064, January 2011.
- [RFC6128]   Begen, A., "RTP Control Protocol (RTCP) Port for Source-Specific Multicast (SSM) Sessions", RFC 6128, February 2011.
- [RFC6189]   Zimmermann, P., Johnston, A., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, April 2011.
- [RFC6193]   Saito, M., Wing, D., and M. Toyama, "Media Description for the Internet Key Exchange Protocol (IKE) in the Session Description Protocol (SDP)", RFC 6193, April 2011.
- [RFC6230]   Boulton, C., Melanchuk, T., and S. McGlashan, "Media Control Channel Framework", RFC 6230, May 2011.
- [RFC6236]   Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.
- [RFC6284]   Begen, A., Wing, D., and T. Van Caenegem, "Port Mapping between Unicast and Multicast RTP Sessions", RFC 6284, June 2011.
- [RFC6285]   Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.
- [RFC6364]   Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, October 2011.
- [RFC6642]   Wu, Q., Xia, F., and R. Even, "RTP Control Protocol (RTCP) Extension for a Third-Party Loss Report", RFC 6642, June 2012.
- [RFC6679]   Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.

- [RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.
- [RFC6773] Phelan, T., Fairhurst, G., and C. Perkins, "DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal", RFC 6773, November 2012.
- [RFC6787] Burnett, D. and S. Shanmugham, "Media Resource Control Protocol Version 2 (MRCPv2)", RFC 6787, November 2012.
- [RTCP-FB] "Session Description Protocol (SDP) RTCP Feedback attributes", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-14>>.
- [SSRC-GROUP]  
"Session Description Protocol (SDP) "ssrc-group" semantics", <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml#sdp-parameters-17>>.
- [T.38] "Procedures for real-time Group 3 facsimile communication over IP networks", <<http://www.itu.int/rec/T-REC-T.38/e>>.

#### Authors' Addresses

Suhas Nandakumar  
Cisco  
170 West Tasman Drive  
San Jose, CA 95134  
USA

Email: [snandaku@cisco.com](mailto:snandaku@cisco.com)

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

Email: [fluffy@iii.ca](mailto:fluffy@iii.ca)



MMUSIC  
Internet-Draft  
Intended status: Standards Track  
Expires: August 29, 2013

M. Petit-Huguenin  
Impedance Mismatch  
A. Keranen  
Ericsson  
February 25, 2013

Using Interactive Connectivity Establishment (ICE) with  
Session Description Protocol (SDP) offer/answer and  
Session Initiation Protocol (SIP)  
draft-petithuguenin-mmusic-ice-sip-sdp-01

Abstract

This document describes how Interactive Connectivity Establishment (ICE) is used with Session Description Protocol (SDP) offer/answer and Session Initiation Protocol (SIP).

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

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

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

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 . . . . .	4
2. Terminology . . . . .	4
3. Sending the Initial Offer . . . . .	4
3.1. Choosing Default Candidates . . . . .	4
3.2. Encoding the SDP . . . . .	5
4. Receiving the Initial Offer . . . . .	6
4.1. Choosing Default Candidates . . . . .	6
4.2. Verifying ICE Support . . . . .	7
4.3. Determining Role . . . . .	7
5. Receipt of the Initial Answer . . . . .	7
5.1. Verifying ICE Support . . . . .	8
6. Performing Connectivity Checks . . . . .	8
7. Concluding ICE . . . . .	8
7.1. Procedures for Full Implementations . . . . .	8
7.1.1. Updating states . . . . .	8
7.2. Freeing Candidates . . . . .	9
7.2.1. Full Implementation Procedures . . . . .	9
8. Grammar . . . . .	9
8.1. "candidate" Attribute . . . . .	9
8.2. "remote-candidates" Attribute . . . . .	11
8.3. "ice-lite" and "ice-mismatch" Attributes . . . . .	12
8.4. "ice-frag" and "ice-pwd" Attributes . . . . .	12
8.5. "ice-options" Attribute . . . . .	13
9. Subsequent Offer/Answer Exchanges . . . . .	13

9.1. Generating the Offer . . . . .	13
9.1.1. Procedures for All Implementations . . . . .	13
9.1.2. Procedures for Full Implementations . . . . .	14
9.1.3. Procedures for Lite Implementations . . . . .	15
9.2. Receiving the Offer and Generating an Answer . . . . .	16
9.2.1. Procedures for All Implementations . . . . .	16
9.2.2. Procedures for Full Implementations . . . . .	17
9.2.3. Procedures for Lite Implementations . . . . .	19
9.3. Updating the Check and Valid Lists . . . . .	20
9.3.1. Procedures for Full Implementations . . . . .	20
9.3.2. Procedures for Lite Implementations . . . . .	21
10. Keepalives . . . . .	21
11. Media Handling . . . . .	21
11.1. Sending Media . . . . .	21
11.1.1. Procedures for All Implementations . . . . .	21
11.2. Receiving Media . . . . .	22
12. Usage with SIP . . . . .	22
12.1. Latency Guidelines . . . . .	22
12.1.1. Offer in INVITE . . . . .	23
12.1.2. Offer in Response . . . . .	24
12.2. SIP Option Tags and Media Feature Tags . . . . .	24
12.3. Interactions with Forking . . . . .	25
12.4. Interactions with Preconditions . . . . .	25
12.5. Interactions with Third Party Call Control . . . . .	25
13. Relationship with ANAT . . . . .	26
14. Setting Ta and RTO for RTP Media Streams . . . . .	26
15. Security Considerations . . . . .	28
15.1. Attacks on the Offer/Answer Exchanges . . . . .	28
15.2. Insider Attacks . . . . .	28
15.2.1. The Voice Hammer Attack . . . . .	28
15.2.2. Interactions with Application Layer Gateways and SIP . . . . .	29
16. IANA Considerations . . . . .	30
16.1. SDP Attributes . . . . .	30
16.1.1. candidate Attribute . . . . .	30
16.1.2. remote-candidates Attribute . . . . .	30
16.1.3. ice-lite Attribute . . . . .	31
16.1.4. ice-mismatch Attribute . . . . .	31
16.1.5. ice-pwd Attribute . . . . .	32
16.1.6. ice-ufrag Attribute . . . . .	32
16.1.7. ice-options Attribute . . . . .	33
16.2. Interactive Connectivity Establishment (ICE) Options Registry . . . . .	33
17. Acknowledgments . . . . .	34
18. References . . . . .	34
18.1. Normative References . . . . .	34
18.2. Informative References . . . . .	35
Appendix A. Examples . . . . .	36
Appendix B. The remote-candidates Attribute . . . . .	37



Appendix C. Why Is the Conflict Resolution Mechanism Needed? . .	38
Appendix D. Why Send an Updated Offer? . . . . .	39
Authors' Addresses . . . . .	40

## 1. Introduction

[NOTE: this version of the document shows merely which parts of the original ICE document could be split to a separate document if the split of SDP is accepted by the WG. Later versions will define the additional procedures needed]

This document describes how Interactive Connectivity Establishment (ICE) is used with Session Description Protocol (SDP) offer/answer and Session Initiation Protocol (SIP). The ICE specification [ICE-BIS] describes procedures that are common to all usages of ICE and this document gives the additional details needed to use ICE with SIP and SDP offer/answer.

Note that ICE is not intended for NAT traversal for SIP, which is assumed to be provided via another mechanism [RFC5626].

## 2. 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].

This document uses the terms defined in [ICE-BIS] and the following:

**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. For the RTP component, the default IP address is in the c line of the SDP, and the port is in the m line. For the RTCP component, it is in the rtcp attribute when present, and when not present, the IP address is in the c line and 1 plus the port is in the m line.

## 3. Sending the Initial Offer

### 3.1. Choosing Default Candidates

A candidate is said to be default if it would be the target of media from a non-ICE peer; that target is called the DEFAULT DESTINATION. If the default candidates are not selected by the ICE algorithm when communicating with an ICE-aware peer, an updated offer/answer will be

required after ICE processing completes in order to "fix up" the SDP so that the default destination for media matches the candidates selected by ICE. If ICE happens to select the default candidates, no updated offer/answer is required.

An agent **MUST** choose a set of candidates, one for each component of each in-use media stream, to be default. A media stream is in-use if it does not have a port of zero (which is used in RFC 3264 to reject a media stream). Consequently, a media stream is in-use even if it is marked as a=inactive [RFC4566] or has a bandwidth value of zero.

It is **RECOMMENDED** that default candidates be chosen based on the likelihood of those candidates to work with the peer that is being contacted. It is **RECOMMENDED** that the default candidates are the relayed candidates (if relayed candidates are available), server reflexive candidates (if server reflexive candidates are available), and finally host candidates.

### 3.2. Encoding the SDP

The process of encoding the SDP is identical between full and lite implementations.

The agent will include an m line for each media stream it wishes to use. The ordering of media streams in the SDP is relevant for ICE. ICE will perform its connectivity checks for the first m line first, and consequently media will be able to flow for that stream first. Agents **SHOULD** place their most important media stream, if there is one, first in the SDP.

There will be a candidate attribute for each candidate for a particular media stream. Section 8 provides detailed rules for constructing this attribute.

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. The username fragment and password are exchanged in the ice-ufrag and ice-pwd attributes, respectively.

If an agent is a lite implementation, it **MUST** include an "a=ice-lite" session-level attribute in its SDP to indicate this. If an agent is a full implementation, it **MUST NOT** include this attribute.

The default candidates are added to the SDP as the default destination for media. For streams based on RTP, this is done by placing the IP address and port of the RTP candidate into the c and m

lines, respectively. If the agent is utilizing RTCP, it MUST encode the RTCP candidate using the a=rtcp attribute as defined in RFC 3605 [RFC3605]. If RTCP is not in use, the agent MUST signal that using b=RS:0 and b=RR:0 as defined in RFC 3556 [RFC3556].

The transport addresses that will be the default destination for media when communicating with non-ICE peers MUST also be present as candidates in one or more a=candidate lines.

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 ice-options attribute.

The following is an example SDP message that includes ICE attributes (lines folded for readability):

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
s=
c=IN IP4 192.0.2.3
t=0 0
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
m=audio 45664 RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr
  10.0.1.1 rport 8998
```

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 of [ICE-BIS], media packets can be sent to a candidate prior to its appearance as the default destination for media in an offer or answer.

#### 4. Receiving the Initial Offer

##### 4.1. Choosing Default Candidates

The process for selecting default candidates at the answerer is identical to the process followed by the offerer, as described in Section 3.1 for full implementations and 4.2 of [ICE-BIS] for lite implementations.

#### 4.2. Verifying ICE Support

The agent will proceed with the ICE procedures defined in [ICE-BIS] and this specification if, for each media stream in the SDP it received, the default destination for each component of that media stream appears in a candidate attribute. For example, in the case of RTP, the IP address and port in the c and m lines, respectively, appear in a candidate attribute and the value in the rtcp attribute appears in a candidate attribute.

If this condition is not met, the agent MUST process the SDP based on normal RFC 3264 procedures, without using any of the ICE mechanisms described in the remainder of this specification with the following exceptions:

1. The agent MUST follow the rules of section 9 of [ICE-BIS], which describe keepalive procedures for all agents.
2. If the agent is not proceeding with ICE because there were a=candidate attributes, but none that matched the default destination of the media stream, the agent MUST include an a=ice-mismatch attribute in its answer.
3. If the default candidates were relayed candidates learned through a TURN server, the agent MUST create permissions in the TURN server for the IP addresses learned from its peer in the SDP it just received. If this is not done, initial packets in the media stream from the peer may be lost.

#### 4.3. Determining Role

In unusual cases, described in Appendix C, it is possible for both agents to mistakenly believe they are controlled or controlling. To resolve this, each agent MUST select a random number, called the tie-breaker, uniformly distributed between 0 and  $(2^{64}) - 1$  (that is, a 64-bit positive integer). This number is used in connectivity checks to detect and repair this case, as described in Section 7.1.2.2 of [ICE-BIS].

#### 5. Receipt of the Initial Answer

When ICE is used with SIP, forking may result in a single offer generating a multiplicity of answers. In that case, ICE proceeds completely in parallel and independently for each answer, treating the combination of its offer and each answer as an independent offer/answer exchange, with its own set of pairs, check lists, states, and so on. The only case in which processing of one pair impacts another is freeing of candidates, discussed below in Section 7.2.

### 5.1. Verifying ICE Support

The logic at the offerer is identical to that of the answerer as described in section 5.1 of [ICE-BIS], with the exception that an offerer would not ever generate a=ice-mismatch attributes in an SDP.

In some cases, the answer may omit a=candidate attributes for the media streams, and instead include an a=ice-mismatch attribute for one or more of the media streams in the SDP. This signals to the offerer that the answerer supports ICE, but that ICE processing was not used for the session because a signaling intermediary modified the default destination for media components without modifying the corresponding candidate attributes. See Section 15.2.2 for a discussion of cases where this can happen. This specification provides no guidance on how an agent should proceed in such a failure case.

## 6. Performing Connectivity Checks

The possibility for role conflicts described in Section 7.2.1.1 of [ICE-BIS] applies to this usage and hence all full agents MUST implement the role conflict repairing mechanism. Also both full and lite agents MUST utilize the ICE-CONTROLLED and ICE-CONTROLLING attributes as described in Section 7.1.2.2 of [ICE-BIS].

## 7. Concluding ICE

Once all of the media streams are completed, the controlling endpoint sends an updated offer if the candidates in the m and c lines for the media stream (called the DEFAULT CANDIDATES) don't match ICE's SELECTED CANDIDATES.

### 7.1. Procedures for Full Implementations

#### 7.1.1. Updating states

Once the state of each check list is Completed, If an agent is controlling, it examines the highest-priority nominated candidate pair for each component of each media stream. If any of those candidate pairs differ from the default candidate pairs in the most recent offer/answer exchange, the controlling agent MUST generate an updated offer as described in Section 9.

## 7.2. Freeing Candidates

### 7.2.1. Full Implementation Procedures

When ICE is used with SIP, and an offer is forked to multiple recipients, ICE proceeds in parallel and independently with each answerer, all using the same local candidates. Once ICE processing has reached the Completed state for all peers for media streams using those candidates, the agent SHOULD wait an additional three seconds, and then it MAY cease responding to checks or generating triggered checks on that candidate. It MAY free the candidate at that time. Freeing of server reflexive candidates is never explicit; it happens by lack of a keepalive. The three-second delay handles cases when aggressive nomination is used, and the selected pairs can quickly change after ICE has completed.

## 8. Grammar

This specification defines seven new SDP attributes -- the "candidate", "remote-candidates", "ice-lite", "ice-mismatch", "ice-ufrag", "ice-pwd", and "ice-options" attributes.

### 8.1. "candidate" Attribute

The candidate attribute is a media-level attribute only. It contains a transport address for a candidate that can be used for connectivity checks.

The syntax of this attribute is defined using Augmented BNF as defined in [RFC5234]:

```
candidate-attribute = "candidate" ":" foundation SP component-id SP
                    transport SP
                    priority SP
                    connection-address SP      ;from RFC 4566
                    port                       ;port from RFC 4566
                    SP cand-type
                    [SP rel-addr]
                    [SP rel-port]
                    *(SP extension-att-name SP
                      extension-att-value)

foundation           = 1*32ice-char
component-id         = 1*5DIGIT
transport            = "UDP" / transport-extension
transport-extension  = token                ; from RFC 3261
priority             = 1*10DIGIT
cand-type            = "typ" SP candidate-types
```

```
candidate-types      = "host" / "srflx" / "prflx" / "relay" / token
rel-addr             = "raddr" SP connection-address
rel-port             = "rport" SP port
extension-att-name   = byte-string ;from RFC 4566
extension-att-value   = byte-string
ice-char             = ALPHA / DIGIT / "+" / "/"
```

This grammar encodes the primary information about a candidate: its IP address, port and transport protocol, and its properties: the foundation, component ID, priority, type, and related transport address:

<connection-address>: is taken from RFC 4566 [RFC4566]. It is the IP address of the candidate, allowing for IPv4 addresses, IPv6 addresses, and fully qualified domain names (FQDNs). When parsing this field, an agent can differentiate an IPv4 address and an IPv6 address by presence of a colon in its value -- the presence of a colon indicates IPv6. An agent MUST ignore candidate lines that include candidates with IP address versions that are not supported or recognized. An IP address SHOULD be used, but an FQDN MAY be used in place of an IP address. In that case, when receiving an offer or answer containing an FQDN in an a=candidate attribute, the FQDN is looked up in the DNS first using an AAAA record (assuming the agent supports IPv6), and if no result is found or the agent only supports IPv4, using an A. If the DNS query returns more than one IP address, one is chosen, and then used for the remainder of ICE processing.

<port>: is also taken from RFC 4566 [RFC4566]. It is the port of the candidate.

<transport>: indicates the transport protocol for the candidate. This specification only defines UDP. However, extensibility is provided to allow for future transport protocols to be used with ICE, such as TCP or the Datagram Congestion Control Protocol (DCCP) [RFC4340].

<foundation>: is composed of 1 to 32 <ice-char>s. It is an identifier that is equivalent for two candidates that are of the same type, share the same base, and come from the same STUN server. The foundation is used to optimize ICE performance in the Frozen algorithm.

<component-id>: is a positive integer between 1 and 256 that identifies the specific component of the media stream for which this is a candidate. It MUST start at 1 and MUST increment by 1 for each component of a particular candidate. For media streams

based on RTP, candidates for the actual RTP media MUST have a component ID of 1, and candidates for RTCP MUST have a component ID of 2. See section 11 in [ICE-BIS] for additional discussion on extending ICE to new media streams.

<priority>: is a positive integer between 1 and  $(2^{31} - 1)$ .

<cand-type>: encodes the type of candidate. This specification defines the values "host", "srflx", "prflx", and "relay" for host, server reflexive, peer reflexive, and relayed candidates, respectively. The set of candidate types is extensible for the future.

<rel-addr> and <rel-port>: convey transport addresses related to the candidate, useful for diagnostics and other purposes. <rel-addr> and <rel-port> MUST be present for server reflexive, peer reflexive, and relayed candidates. If a candidate is server or peer reflexive, <rel-addr> and <rel-port> are equal to the base for that server or peer reflexive candidate. If the candidate is relayed, <rel-addr> and <rel-port> is equal to the mapped address in the Allocate response that provided the client with that relayed candidate (see section Appendix B.3 of [ICE-BIS] for a discussion of its purpose). If the candidate is a host candidate, <rel-addr> and <rel-port> MUST be omitted.

The candidate attribute can itself be extended. The grammar allows for new name/value pairs to be added at the end of the attribute. An implementation MUST ignore any name/value pairs it doesn't understand.

## 8.2. "remote-candidates" Attribute

The syntax of the "remote-candidates" attribute is defined using Augmented BNF as defined in RFC 5234 [RFC5234]. The remote-candidates attribute is a media-level attribute only.

```
remote-candidate-att = "remote-candidates" ":" remote-candidate
                      0*(SP remote-candidate)
remote-candidate = component-ID SP connection-address SP port
```

The attribute contains a connection-address and port for each component. The ordering of components is irrelevant. However, a value MUST be present for each component of a media stream. This attribute MUST be included in an offer by a controlling agent for a media stream that is Completed, and MUST NOT be included in any other case.



### 8.3. "ice-lite" and "ice-mismatch" Attributes

The syntax of the "ice-lite" and "ice-mismatch" attributes, both of which are flags, is:

```
ice-lite           = "ice-lite"
ice-mismatch       = "ice-mismatch"
```

"ice-lite" is a session-level attribute only, and indicates that an agent is a lite implementation. "ice-mismatch" is a media-level attribute only, and when present in an answer, indicates that the offer arrived with a default destination for a media component that didn't have a corresponding candidate attribute.

### 8.4. "ice-ufrag" and "ice-pwd" Attributes

The "ice-ufrag" and "ice-pwd" attributes convey the username fragment and password used by ICE for message integrity. Their syntax is:

```
ice-pwd-att        = "ice-pwd" ":" password
ice-ufrag-att      = "ice-ufrag" ":" ufrag
password           = 22*256ice-char
ufrag              = 4*256ice-char
```

The "ice-pwd" and "ice-ufrag" attributes can appear at either the session-level or media-level. When present in both, the value in the media-level takes precedence. Thus, the value at the session-level is effectively a default that applies to all media streams, unless overridden by a media-level value. Whether present at the session or media-level, there MUST be an ice-pwd and ice-ufrag attribute for each media stream. If two media streams have identical ice-ufrag's, they MUST have identical ice-pwd's.

The ice-ufrag and ice-pwd attributes MUST be chosen randomly at the beginning of a session. The ice-ufrag attribute MUST contain at least 24 bits of randomness, and the ice-pwd attribute MUST contain at least 128 bits of randomness. This means that the ice-ufrag attribute will be at least 4 characters long, and the ice-pwd at least 22 characters long, since the grammar for these attributes allows for 6 bits of randomness per character. The attributes MAY be longer than 4 and 22 characters, respectively, of course, up to 256 characters. The upper limit allows for buffer sizing in implementations. Its large upper limit allows for increased amounts of randomness to be added over time.

### 8.5. "ice-options" Attribute

The "ice-options" attribute is a session-level attribute. It contains a series of tokens that identify the options supported by the agent. Its grammar is:

```
ice-options          = "ice-options" ":" ice-option-tag
                      0*(SP ice-option-tag)
ice-option-tag       = 1*ice-char
```

## 9. Subsequent Offer/Answer Exchanges

Either agent MAY generate a subsequent offer at any time allowed by RFC 3264 [RFC3264]. The rules in Section 7 will cause the controlling agent to send an updated offer at the conclusion of ICE processing when ICE has selected different candidate pairs from the default pairs. This section defines rules for construction of subsequent offers and answers.

Should a subsequent offer be rejected, ICE processing continues as if the subsequent offer had never been made.

### 9.1. Generating the Offer

#### 9.1.1. Procedures for All Implementations

##### 9.1.1.1. ICE Restarts

An agent MAY restart ICE processing for an existing media stream. An ICE restart, as the name implies, will cause all previous states of ICE processing to be flushed and checks to start anew. The only difference between an ICE restart and a brand new media session is that, during the restart, media can continue to be sent to the previously validated pair.

An agent MUST restart ICE for a media stream if:

- o The offer is being generated for the purposes of changing the target of the media stream. In other words, if an agent wants to generate an updated offer that, had ICE not been in use, would result in a new value for the destination of a media component.
- o An agent is changing its implementation level. This typically only happens in third party call control use cases, where the entity performing the signaling is not the entity receiving the media, and it has changed the target of media mid-session to another entity that has a different ICE implementation.

These rules imply that setting the IP address in the c line to 0.0.0.0 will cause an ICE restart. Consequently, ICE implementations MUST NOT utilize this mechanism for call hold, and instead MUST use a=inactive and a=sendonly as described in [RFC3264].

To restart ICE, an agent MUST change both the ice-pwd and the ice-ufrag for the media stream in an offer. Note that it is permissible to use a session-level attribute in one offer, but to provide the same ice-pwd or ice-ufrag as a media-level attribute in a subsequent offer. This is not a change in password, just a change in its representation, and does not cause an ICE restart.

An agent sets the rest of the fields in the SDP for this media stream as it would in an initial offer of this media stream (see Section 3.2). Consequently, the set of candidates MAY include some, none, or all of the previous candidates for that stream and MAY include a totally new set of candidates.

#### 9.1.1.2. Removing a Media Stream

If an agent removes a media stream by setting its port to zero, it MUST NOT include any candidate attributes for that media stream and SHOULD NOT include any other ICE-related attributes defined in Section 8 for that media stream.

#### 9.1.1.3. Adding a Media Stream

If an agent wishes to add a new media stream, it sets the fields in the SDP for this media stream as if this was an initial offer for that media stream (see Section 3.2). This will cause ICE processing to begin for this media stream.

#### 9.1.2. Procedures for Full Implementations

This section describes additional procedures for full implementations, covering existing media streams.

The username fragments, password, and implementation level MUST remain the same as used previously. If an agent needs to change one of these, it MUST restart ICE for that media stream.

Additional behavior depends on the state ICE processing for that media stream.

##### 9.1.2.1. Existing Media Streams with ICE Running

If an agent generates an updated offer including a media stream that was previously established, and for which ICE checks are in the Running state, the agent follows the procedures defined here.

An agent MUST include candidate attributes for all local candidates it had signaled previously for that media stream. The properties of that candidate as signaled in SDP -- the priority, foundation, type, and related transport address -- SHOULD remain the same. The IP address, port, and transport protocol, which fundamentally identify that candidate, MUST remain the same (if they change, it would be a new candidate). The component ID MUST remain the same. The agent MAY include additional candidates it did not offer previously, but which it has gathered since the last offer/answer exchange, including peer reflexive candidates.

The agent MAY change the default destination for media. As with initial offers, there MUST be a set of candidate attributes in the offer matching this default destination.

#### 9.1.2.2. Existing Media Streams with ICE Completed

If an agent generates an updated offer including a media stream that was previously established, and for which ICE checks are in the Completed state, the agent follows the procedures defined here.

The default destination for media (i.e., the values of the IP addresses and ports in the m and c lines used for that media stream) MUST be the local candidate from the highest-priority nominated pair in the valid list for each component. This "fixes" the default destination for media to equal the destination ICE has selected for media.

The agent MUST include candidate attributes for candidates matching the default destination for each component of the media stream, and MUST NOT include any other candidates.

In addition, if the agent is controlling, it MUST include the a=remote-candidates attribute for each media stream whose check list is in the Completed state. The attribute contains the remote candidates from the highest-priority nominated pair in the valid list for each component of that media stream. It is needed to avoid a race condition whereby the controlling agent chooses its pairs, but the updated offer beats the connectivity checks to the controlled agent, which doesn't even know these pairs are valid, let alone selected. See Appendix B for elaboration on this race condition.

#### 9.1.3. Procedures for Lite Implementations

#### 9.1.3.1. Existing Media Streams with ICE Running

This section describes procedures for lite implementations for existing streams for which ICE is running.

A lite implementation **MUST** include all of its candidates for each component of each media stream in an `a=candidate` attribute in any subsequent offer. These candidates are formed identically to the procedures for initial offers, as described in section 4.2 of [ICE-BIS].

A lite implementation **MUST NOT** add additional host candidates in a subsequent offer. If an agent needs to offer additional candidates, it **MUST** restart ICE.

The username fragments, password, and implementation level **MUST** remain the same as used previously. If an agent needs to change one of these, it **MUST** restart ICE for that media stream.

#### 9.1.3.2. Existing Media Streams with ICE Completed

If ICE has completed for a media stream, the default destination for that media stream **MUST** be set to the remote candidate of the candidate pair for that component in the valid list. For a lite implementation, there is always just a single candidate pair in the valid list for each component of a media stream. Additionally, the agent **MUST** include a candidate attribute for each default destination.

Additionally, if the agent is controlling (which only happens when both agents are lite), the agent **MUST** include the `a=remote-candidates` attribute for each media stream. The attribute contains the remote candidates from the candidate pairs in the valid list (one pair for each component of each media stream).

### 9.2. Receiving the Offer and Generating an Answer

#### 9.2.1. Procedures for All Implementations

When receiving a subsequent offer within an existing session, an agent **MUST** reapply the verification procedures in Section 4.2 without regard to the results of verification from any previous offer/answer exchanges. Indeed, it is possible that a previous offer/answer exchange resulted in ICE not being used, but it is used as a consequence of a subsequent exchange.

##### 9.2.1.1. Detecting ICE Restart

If the offer contained a change in the a=ice-ufrag or a=ice-pwd attributes compared to the previous SDP from the peer, it indicates that ICE is restarting for this media stream. If all media streams are restarting, then ICE is restarting overall.

If ICE is restarting for a media stream:

- o The agent **MUST** change the a=ice-ufrag and a=ice-pwd attributes in the answer.
- o The agent **MAY** change its implementation level in the answer.

An agent sets the rest of the fields in the SDP for this media stream as it would in an initial answer to this media stream (see Section 3.2). Consequently, the set of candidates **MAY** include some, none, or all of the previous candidates for that stream and **MAY** include a totally new set of candidates.

#### 9.2.1.2. New Media Stream

If the offer contains a new media stream, the agent sets the fields in the answer as if it had received an initial offer containing that media stream (see Section 3.2). This will cause ICE processing to begin for this media stream.

#### 9.2.1.3. Removed Media Stream

If an offer contains a media stream whose port is zero, the agent **MUST NOT** include any candidate attributes for that media stream in its answer and **SHOULD NOT** include any other ICE-related attributes defined in Section 8 for that media stream.

#### 9.2.2. Procedures for Full Implementations

Unless the agent has detected an ICE restart from the offer, the username fragments, password, and implementation level **MUST** remain the same as used previously. If an agent needs to change one of these it **MUST** restart ICE for that media stream by generating an offer; ICE cannot be restarted in an answer.

Additional behaviors depend on the state of ICE processing for that media stream.

##### 9.2.2.1. Existing Media Streams with ICE Running and no remote-candidates

If ICE is running for a media stream, and the offer for that media stream lacked the remote-candidates attribute, the rules for

construction of the answer are identical to those for the offerer as described in Section 9.1.2.1.

#### 9.2.2.2. Existing Media Streams with ICE Completed and no remote-candidates

If ICE is Completed for a media stream, and the offer for that media stream lacked the remote-candidates attribute, the rules for construction of the answer are identical to those for the offerer as described in Section 9.1.2.2, except that the answerer MUST NOT include the a=remote-candidates attribute in the answer.

#### 9.2.2.3. Existing Media Streams and remote-candidates

A controlled agent will receive an offer with the a=remote-candidates attribute for a media stream when its peer has concluded ICE processing for that media stream. This attribute is present in the offer to deal with a race condition between the receipt of the offer, and the receipt of the Binding response that tells the answerer the candidate that will be selected by ICE. See Appendix B for an explanation of this race condition. Consequently, processing of an offer with this attribute depends on the winner of the race.

The agent forms a candidate pair for each component of the media stream by:

- o Setting the remote candidate equal to the offerer's default destination for that component (e.g., the contents of the m and c lines for RTP, and the a=rtcp attribute for RTCP)
- o Setting the local candidate equal to the transport address for that same component in the a=remote-candidates attribute in the offer.

The agent then sees if each of these candidate pairs is present in the valid list. If a particular pair is not in the valid list, the check has "lost" the race. Call such a pair a "losing pair".

The agent finds all the pairs in the check list whose remote candidates equal the remote candidate in the losing pair:

- o If none of the pairs are In-Progress, and at least one is Failed, it is most likely that a network failure, such as a network partition or serious packet loss, has occurred. The agent SHOULD generate an answer for this media stream as if the remote-candidates attribute had not been present, and then restart ICE for this stream.

- o If at least one of the pairs is In-Progress, the agent SHOULD wait for those checks to complete, and as each completes, redo the processing in this section until there are no losing pairs.

Once there are no losing pairs, the agent can generate the answer. It MUST set the default destination for media to the candidates in the remote-candidates attribute from the offer (each of which will now be the local candidate of a candidate pair in the valid list). It MUST include a candidate attribute in the answer for each candidate in the remote-candidates attribute in the offer.

### 9.2.3. Procedures for Lite Implementations

If the received offer contains the remote-candidates attribute for a media stream, the agent forms a candidate pair for each component of the media stream by:

- o Setting the remote candidate equal to the offerer's default destination for that component (e.g., the contents of the m and c lines for RTP, and the a=rtcp attribute for RTCP).
- o Setting the local candidate equal to the transport address for that same component in the a=remote-candidates attribute in the offer.

It then places those candidates into the Valid list for the media stream. The state of ICE processing for that media stream is set to Completed.

Furthermore, if the agent believed it was controlling, but the offer contained the remote-candidates attribute, both agents believe they are controlling. In this case, both would have sent updated offers around the same time. However, the signaling protocol carrying the offer/answer exchanges will have resolved this glare condition, so that one agent is always the 'winner' by having its offer received before its peer has sent an offer. The winner takes the role of controlled, so that the loser (the answerer under consideration in this section) MUST change its role to controlled. Consequently, if the agent was going to send an updated offer since, based on the rules in section 8.2 of [ICE-BIS], it was controlling, it no longer needs to.

Besides the potential role change, change in the Valid list, and state changes, the construction of the answer is performed identically to the construction of an offer as described in Section 9.1.3.



### 9.3. Updating the Check and Valid Lists

#### 9.3.1. Procedures for Full Implementations

##### 9.3.1.1. ICE Restarts

The agent MUST remember the highest-priority nominated pairs in the Valid list for each component of the media stream, called the previous selected pairs, prior to the restart. The agent will continue to send media using these pairs, as described in Section 11.1. Once these destinations are noted, the agent MUST flush the valid and check lists, and then recompute the check list and its states as described in section 6.3 of [ICE-BIS].

##### 9.3.1.2. New Media Stream

If the offer/answer exchange added a new media stream, the agent MUST create a new check list for it (and an empty Valid list to start of course), as described in section 6.3 of [ICE-BIS].

##### 9.3.1.3. Removed Media Stream

If the offer/answer exchange removed a media stream, or an answer rejected an offered media stream, an agent MUST flush the Valid list for that media stream. It MUST terminate any STUN transactions in progress for that media stream. An agent MUST remove the check list for that media stream and cancel any pending ordinary checks for it.

##### 9.3.1.4. ICE Continuing for Existing Media Stream

The valid list is not affected by an updated offer/answer exchange unless ICE is restarting.

If an agent is in the Running state for that media stream, the check list is updated (the check list is irrelevant if the state is completed). To do that, the agent recomputes the check list using the procedures described in section 6.3 of [ICE-BIS]. If a pair on the new check list was also on the previous check list, and its state was Waiting, In-Progress, Succeeded, or Failed, its state is copied over. Otherwise, its state is set to Frozen.

If none of the check lists are active (meaning that the pairs in each check list are Frozen), the full-mode agent sets the first pair in the check list for the first media stream to Waiting, and then sets the state of all other pairs in that check list for the same component ID and with the same foundation to Waiting as well.

Next, the agent goes through each check list, starting with the highest-priority pair. If a pair has a state of Succeeded, and it has a component ID of 1, then all Frozen pairs in the same check list with the same foundation whose component IDs are not 1 have their state set to Waiting. If, for a particular check list, there are pairs for each component of that media stream in the Succeeded state, the agent moves the state of all Frozen pairs for the first component of all other media streams (and thus in different check lists) with the same foundation to Waiting.

#### 9.3.2. Procedures for Lite Implementations

If ICE is restarting for a media stream, the agent MUST start a new Valid list for that media stream. It MUST remember the pairs in the previous Valid list for each component of the media stream, called the previous selected pairs, and continue to send media there as described in Section 11.1. The state of ICE processing for each media stream MUST change to Running, and the state of ICE processing MUST change to Running.

### 10. Keepalives

The keepalives MUST be sent regardless of whether the media stream is currently inactive, sendonly, recvonly, or sendrecv, and regardless of the presence or value of the bandwidth attribute. An agent can determine that its peer supports ICE by the presence of a=candidate attributes for each media session.

### 11. Media Handling

#### 11.1. Sending Media

Note that the selected pair for a component of a media stream may not equal the default pair for that same component from the most recent offer/answer exchange. When this happens, the selected pair is used for media, not the default pair. When ICE first completes, if the selected pairs aren't a match for the default pairs, the controlling agent sends an updated offer/answer exchange to remedy this disparity. However, until that updated offer arrives, there will not be a match. Furthermore, in very unusual cases, the default candidates in the updated offer/answer will not be a match.

##### 11.1.1. 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.

## 11.2. 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.

## 12. Usage with SIP

### 12.1. Latency Guidelines

ICE requires a series of STUN-based connectivity checks to take place between endpoints. These checks start from the answerer on generation of its answer, and start from the offerer when it receives the answer. These checks can take time to complete, and as such, the selection of messages to use with offers and answers can affect perceived user latency. Two latency figures are of particular interest. These are the post-pickup delay and the post-dial delay. The post-pickup delay refers to the time between when a user "answers the phone" and when any speech they utter can be delivered to the caller. The post-dial delay refers to the time between when a user enters the destination address for the user and ringback begins as a consequence of having successfully started ringing the phone of the called party.

Two cases can be considered -- one where the offer is present in the initial INVITE and one where it is in a response.

#### 12.1.1.1. Offer in INVITE

To reduce post-dial delays, it is RECOMMENDED that the caller begin gathering candidates prior to actually sending its initial INVITE. This can be started upon user interface cues that a call is pending, such as activity on a keypad or the phone going off-hook.

If an offer is received in an INVITE request, the answerer SHOULD begin to gather its candidates on receipt of the offer and then generate an answer in a provisional response once it has completed that process. ICE requires that a provisional response with an SDP be transmitted reliably. This can be done through the existing Provisional Response Acknowledgment (PRACK) mechanism [RFC3262] or through an optimization that is specific to ICE. With this optimization, provisional responses containing an SDP answer that begins ICE processing for one or more media streams can be sent reliably without RFC 3262. To do this, the agent retransmits the provisional response with the exponential backoff timers described in RFC 3262. Retransmits MUST cease on receipt of a STUN Binding request for one of the media streams signaled in that SDP (because receipt of a Binding request indicates the offerer has received the answer) or on transmission of the answer in a 2xx response. If the peer agent is lite, there will never be a STUN Binding request. In such a case, the agent MUST cease retransmitting the 18x after sending it four times (ICE will actually work even if the peer never receives the 18x; however, experience has shown that sending it is important for middleboxes and firewall traversal). If no Binding request is received prior to the last retransmit, the agent does not consider the session terminated. Despite the fact that the provisional response will be delivered reliably, the rules for when an agent can send an updated offer or answer do not change from those specified in RFC 3262. Specifically, if the INVITE contained an offer, the same answer appears in all of the 1xx and in the 2xx response to the INVITE. Only after that 2xx has been sent can an updated offer/answer exchange occur. This optimization SHOULD NOT be used if both agents support PRACK. Note that the optimization is very specific to provisional response carrying answers that start ICE processing; it is not a general technique for 1xx reliability.

Alternatively, an agent MAY delay sending an answer until the 200 OK; however, this results in a poor user experience and is NOT RECOMMENDED.

Once the answer has been sent, the agent SHOULD begin its connectivity checks. Once candidate pairs for each component of a

media stream enter the valid list, the answerer can begin sending media on that media stream.

However, prior to this point, any media that needs to be sent towards the caller (such as SIP early media [RFC3960]) MUST NOT be transmitted. For this reason, implementations SHOULD delay alerting the called party until candidates for each component of each media stream have entered the valid list. In the case of a PSTN gateway, this would mean that the setup message into the PSTN is delayed until this point. Doing this increases the post-dial delay, but has the effect of eliminating 'ghost rings'. Ghost rings are cases where the called party hears the phone ring, picks up, but hears nothing and cannot be heard. This technique works without requiring support for, or usage of, preconditions [RFC3312], since it's a localized decision. It also has the benefit of guaranteeing that not a single packet of media will get clipped, so that post-pickup delay is zero. If an agent chooses to delay local alerting in this way, it SHOULD generate a 180 response once alerting begins.

#### 12.1.2. Offer in Response

In addition to uses where the offer is in an INVITE, and the answer is in the provisional and/or 200 OK response, ICE works with cases where the offer appears in the response. In such cases, which are common in third party call control [RFC3725], ICE agents SHOULD generate their offers in a reliable provisional response (which MUST utilize RFC 3262), and not alert the user on receipt of the INVITE. The answer will arrive in a PRACK. This allows for ICE processing to take place prior to alerting, so that there is no post-pickup delay, at the expense of increased call setup delays. Once ICE completes, the callee can alert the user and then generate a 200 OK when they answer. The 200 OK would contain no SDP, since the offer/answer exchange has completed.

Alternatively, agents MAY place the offer in a 2xx instead (in which case the answer comes in the ACK). When this happens, the callee will alert the user on receipt of the INVITE, and the ICE exchanges will take place only after the user answers. This has the effect of reducing call setup delay, but can cause substantial post-pickup delays and media clipping.

#### 12.2. SIP Option Tags and Media Feature Tags

[RFC5768] specifies a SIP option tag and media feature tag for usage with ICE. ICE implementations using SIP SHOULD support this specification, which uses a feature tag in registrations to facilitate interoperability through signaling intermediaries.

### 12.3. Interactions with Forking

ICE interacts very well with forking. Indeed, ICE fixes some of the problems associated with forking. Without ICE, when a call forks and the caller receives multiple incoming media streams, it cannot determine which media stream corresponds to which callee.

With ICE, this problem is resolved. The connectivity checks which occur prior to transmission of media carry username fragments, which in turn are correlated to a specific callee. Subsequent media packets that arrive on the same candidate pair as the connectivity check will be associated with that same callee. Thus, the caller can perform this correlation as long as it has received an answer.

### 12.4. Interactions with Preconditions

Quality of Service (QoS) preconditions, which are defined in RFC 3312 [RFC3312] and RFC 4032 [RFC4032], apply only to the transport addresses listed as the default targets for media in an offer/answer. If ICE changes the transport address where media is received, this change is reflected in an updated offer that changes the default destination for media to match ICE's selection. As such, it appears like any other re-INVITE would, and is fully treated in RFCs 3312 and 4032, which apply without regard to the fact that the destination for media is changing due to ICE negotiations occurring "in the background".

Indeed, an agent SHOULD NOT indicate that QoS preconditions have been met until the checks have completed and selected the candidate pairs to be used for media.

ICE also has (purposeful) interactions with connectivity preconditions [RFC5898]. Those interactions are described there. Note that the procedures described in Section 12.1 describe their own type of "preconditions", albeit with less functionality than those provided by the explicit preconditions in [RFC5898].

### 12.5. Interactions with Third Party Call Control

ICE works with Flows I, III, and IV as described in [RFC3725]. Flow I works without the controller supporting or being aware of ICE. Flow IV will work as long as the controller passes along the ICE attributes without alteration. Flow II is fundamentally incompatible with ICE; each agent will believe itself to be the answerer and thus never generate a re-INVITE.

The flows for continued operation, as described in Section 7 of RFC 3725, require additional behavior of ICE implementations to support.

In particular, if an agent receives a mid-dialog re-INVITE that contains no offer, it MUST restart ICE for each media stream and go through the process of gathering new candidates. Furthermore, that list of candidates SHOULD include the ones currently being used for media.

### 13. Relationship with ANAT

RFC 4091 [RFC4091], the Alternative Network Address Types (ANAT) Semantics for the SDP grouping framework, and RFC 4092 [RFC4092], its usage with SIP, define a mechanism for indicating that an agent can support both IPv4 and IPv6 for a media stream, and it does so by including two m lines, one for v4 and one for v6. This is similar to ICE, which allows for an agent to indicate multiple transport addresses using the candidate attribute. However, ANAT relies on static selection to pick between choices, rather than a dynamic connectivity check used by ICE.

This specification deprecates RFC 4091 and RFC 4092. Instead, agents wishing to support dual-stack will utilize ICE.

### 14. Setting Ta and RTO for RTP Media Streams

During the gathering phase of ICE (section 4.1.1 [ICE-BIS]) and while ICE is performing connectivity checks (section 7 [ICE-BIS]), 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 with a real-time media stream (such as RTP). When ICE is used for a stream with a known maximum bandwidth, the following computation MAY be followed to rate-control the ICE exchanges.

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

$$Ta = \text{MAX} \left( 20\text{ms}, \frac{1}{k} \right)$$

$$\frac{1}{k} = \frac{1}{\frac{stun\_packet\_size}{rtp\_packet\_size} * rtp\_ptime}$$

----  
i=1

where  $k$  is the number of media streams. During the gathering phase,  $T_a$  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  $T_a$  to pace the connectivity checks. In that case, the value of  $T_a$  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:

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

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:

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

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 section B.1 of [ICE-BIS] 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.

## 15. Security Considerations

### 15.1. Attacks on the Offer/Answer Exchanges

An attacker that can modify or disrupt the offer/answer exchanges themselves can readily launch a variety of attacks with ICE. They could direct media to a target of a DoS attack, they could insert themselves into the media stream, and so on. These are similar to the general security considerations for offer/answer exchanges, and the security considerations in RFC 3264 [RFC3264] apply. These require techniques for message integrity and encryption for offers and answers, which are satisfied by the SIPS mechanism [RFC3261] when SIP is used. As such, the usage of SIPS with ICE is RECOMMENDED.

### 15.2. Insider Attacks

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

#### 15.2.1. The Voice Hammer Attack

The voice hammer attack is an amplification attack. In this attack, the attacker initiates sessions to other agents, and maliciously includes the IP address and port of a DoS target as the destination for media traffic signaled in the SDP. This causes substantial amplification; a single offer/answer exchange can create a continuing flood of media packets, possibly at high rates (consider video sources). This attack is not specific to ICE, but ICE can help provide remediation.

Specifically, if ICE is used, the agent receiving the malicious SDP will first perform connectivity checks to the target of media before sending media there. If this target is a third-party host, the checks will not succeed, and media is never sent.

Unfortunately, ICE doesn't help if its not used, in which case an attacker could simply send the offer without the ICE parameters. However, in environments where the set of clients is known, and is limited to ones that support ICE, the server can reject any offers or answers that don't indicate ICE support.

#### 15.2.2. Interactions with Application Layer Gateways and SIP

Application Layer Gateways (ALGs) are functions present in a NAT device that inspect the contents of packets and modify them, in order to facilitate NAT traversal for application protocols. Session Border Controllers (SBCs) are close cousins of ALGs, but are less transparent since they actually exist as application layer SIP intermediaries. ICE has interactions with SBCs and ALGs.

If an ALG is SIP aware but not ICE aware, ICE will work through it as long as the ALG correctly modifies the SDP. A correct ALG implementation behaves as follows:

- o The ALG does not modify the m and c lines or the rtcp attribute if they contain external addresses.
- o If the m and c lines contain internal addresses, the modification depends on the state of the ALG:

If the ALG already has a binding established that maps an external port to an internal IP address and port matching the values in the m and c lines or rtcp attribute, the ALG uses that binding instead of creating a new one.

If the ALG does not already have a binding, it creates a new one and modifies the SDP, rewriting the m and c lines and rtcp attribute.

Unfortunately, many ALGs are known to work poorly in these corner cases. ICE does not try to work around broken ALGs, as this is outside the scope of its functionality. ICE can help diagnose these conditions, which often show up as a mismatch between the set of candidates and the m and c lines and rtcp attributes. The ice-mismatch attribute is used for this purpose.

ICE works best through ALGs when the signaling is run over TLS. This prevents the ALG from manipulating the SDP messages and interfering

with ICE operation. Implementations that are expected to be deployed behind ALGs SHOULD provide for TLS transport of the SDP.

If an SBC is SIP aware but not ICE aware, the result depends on the behavior of the SBC. If it is acting as a proper Back-to-Back User Agent (B2BUA), the SBC will remove any SDP attributes it doesn't understand, including the ICE attributes. Consequently, the call will appear to both endpoints as if the other side doesn't support ICE. This will result in ICE being disabled, and media flowing through the SBC, if the SBC has requested it. If, however, the SBC passes the ICE attributes without modification, yet modifies the default destination for media (contained in the m and c lines and rtcp attribute), this will be detected as an ICE mismatch, and ICE processing is aborted for the call. It is outside of the scope of ICE for it to act as a tool for "working around" SBCs. If one is present, ICE will not be used and the SBC techniques take precedence.

## 16. IANA Considerations

### 16.1. SDP Attributes

Original ICE specification defined seven new SDP attributes per the procedures of Section 8.2.4 of [RFC4566]. The registration information is reproduced here.

#### 16.1.1. candidate Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: candidate

Long Form: candidate

Type of Attribute: media-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and provides one of many possible candidate addresses for communication. These addresses are validated with an end-to-end connectivity check using Session Traversal Utilities for NAT (STUN).

Appropriate Values: See Section 8 of RFC XXXX.

#### 16.1.2. remote-candidates Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: remote-candidates

Long Form: remote-candidates

Type of Attribute: media-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and provides the identity of the remote candidates that the offerer wishes the answerer to use in its answer.

Appropriate Values: See Section 8 of RFC XXXX.

#### 16.1.3. ice-lite Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-lite

Long Form: ice-lite

Type of Attribute: session-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and indicates that an agent has the minimum functionality required to support ICE inter-operation with a peer that has a full implementation.

Appropriate Values: See Section 8 of RFC XXXX.

#### 16.1.4. ice-mismatch Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-mismatch

Long Form: ice-mismatch

Type of Attribute: session-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and indicates that an agent is ICE capable, but did not proceed with ICE due to a mismatch of candidates with the default destination for media signaled in the SDP.

Appropriate Values: See Section 8 of RFC XXXX.

#### 16.1.5. ice-pwd Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-pwd

Long Form: ice-pwd

Type of Attribute: session- or media-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and provides the password used to protect STUN connectivity checks.

Appropriate Values: See Section 8 of RFC XXXX.

#### 16.1.6. ice-ufrag Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-ufrag

Long Form: ice-ufrag

Type of Attribute: session- or media-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and provides the fragments used to construct the username in STUN connectivity checks.

Appropriate Values: See Section 8 of RFC XXXX.

#### 16.1.7. ice-options Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-options

Long Form: ice-options

Type of Attribute: session-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and indicates the ICE options or extensions used by the agent.

Appropriate Values: See Section 8 of RFC XXXX.

#### 16.2. Interactive Connectivity Establishment (ICE) Options Registry

IANA maintains a registry for ice-options identifiers under the Specification Required policy as defined in "Guidelines for Writing an IANA Considerations Section in RFCs" [RFC5226].

ICE options are of unlimited length according to the syntax in Section 8.5; however, they are RECOMMENDED to be no longer than 20 characters. This is to reduce message sizes and allow for efficient parsing.

A registration request MUST include the following information:

- o The ICE option identifier to be registered
- o Name, Email, and Address of a contact person for the registration
- o Organization or individuals having the change control
- o Short description of the ICE extension to which the option relates
- o Reference(s) to the specification defining the ICE option and the related extensions

## 17. Acknowledgments

A large part of the text in this document was taken from RFC 5245, authored by Jonathan Rosenberg.

Some of the text in this document was taken from RFC 6336, authored by Magnus Westerlund and Colin Perkins.

## 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.
- [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.
- [RFC3262] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3312] Camarillo, G., Marshall, W., and J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)", RFC 3312, 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.
- [RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.
- [RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.
- [RFC4032] Camarillo, G. and P. Kyzivat, "Update to the Session Initiation Protocol (SIP) Preconditions Framework", RFC 4032, 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.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5768] Rosenberg, J., "Indicating Support for Interactive Connectivity Establishment (ICE) in the Session Initiation Protocol (SIP)", RFC 5768, April 2010.
- [ICE-BIS] Keranen, A. and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-keranen-mmusic-rfc5245bis-01 (work in progress), February 2013.

## 18.2. Informative References

- [RFC3725] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.
- [RFC3960] Camarillo, G. and H. Schulzrinne, "Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP)", RFC 3960, December 2004.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.



[RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

[RFC5898] Andreassen, F., Camarillo, G., Oran, D., and D. Wing, "Connectivity Preconditions for Session Description Protocol (SDP) Media Streams", RFC 5898, July 2010.

## Appendix A. Examples

For the example shown in Section 13 of [ICE-BIS] the resulting offer (message 5) encoded in SDP looks like:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 $L-PRIV-1.IP
s=
c=IN IP4 $NAT-PUB-1.IP
t=0 0
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
m=audio $NAT-PUB-1.PORT RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706431 $L-PRIV-1.IP $L-PRIV-1.PORT typ host
a=candidate:2 1 UDP 1694498815 $NAT-PUB-1.IP $NAT-PUB-1.PORT typ
  srflx raddr $L-PRIV-1.IP rport $L-PRIV-1.PORT
```

The offer, with the variables replaced with their values, will look like (lines folded for clarity):

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
s=
c=IN IP4 192.0.2.3
t=0 0
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
m=audio 45664 RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr
  10.0.1.1 rport 8998
```

The resulting answer looks like:

```
v=0
o=bob 2808844564 2808844564 IN IP4 $R-PUB-1.IP
s=
c=IN IP4 $R-PUB-1.IP
t=0 0
a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
a=ice-ufrag:9uB6
m=audio $R-PUB-1.PORT RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706431 $R-PUB-1.IP $R-PUB-1.PORT typ host
```

With the variables filled in:

```
v=0
o=bob 2808844564 2808844564 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
a=ice-ufrag:9uB6
m=audio 3478 RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706431 192.0.2.1 3478 typ host
```

## Appendix B. The remote-candidates Attribute

The `a=remote-candidates` attribute exists to eliminate a race condition between the updated offer and the response to the STUN Binding request that moved a candidate into the Valid list. This race condition is shown in Figure 1. On receipt of message 4, agent L adds a candidate pair to the valid list. If there was only a single media stream with a single component, agent L could now send an updated offer. However, the check from agent R has not yet generated a response, and agent R receives the updated offer (message 7) before getting the response (message 9). Thus, it does not yet know that this particular pair is valid. To eliminate this condition, the actual candidates at R that were selected by the offerer (the remote candidates) are included in the offer itself, and the answerer delays its answer until those pairs validate.

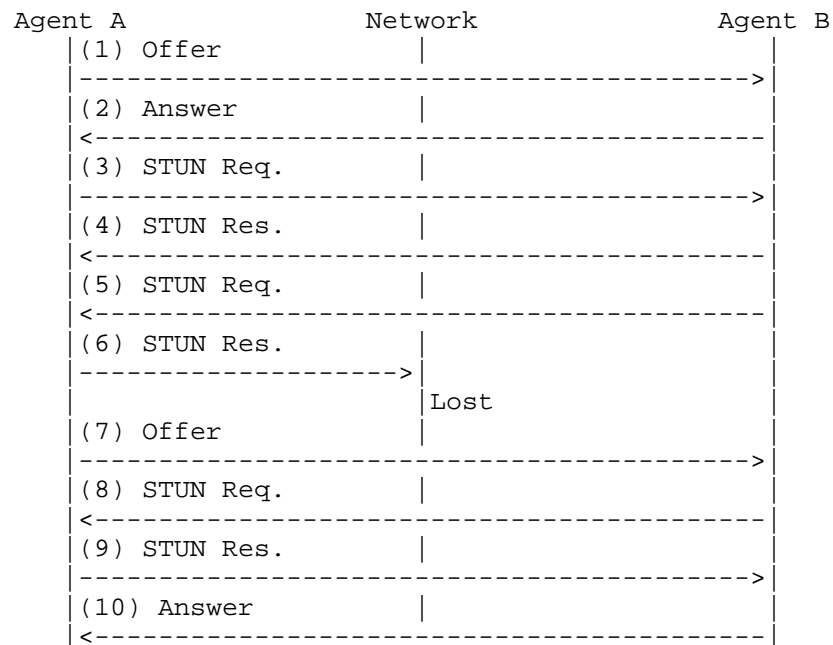
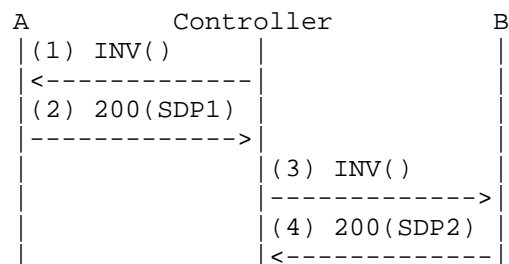


Figure 1: Race Condition Flow

#### Appendix C. Why Is the Conflict Resolution Mechanism Needed?

When ICE runs between two peers, one agent acts as controlled, and the other as controlling. Rules are defined as a function of implementation type and offerer/answerer to determine who is controlling and who is controlled. However, the specification mentions that, in some cases, both sides might believe they are controlling, or both sides might believe they are controlled. How can this happen?

The condition when both agents believe they are controlled shows up in third party call control cases. Consider the following flow:



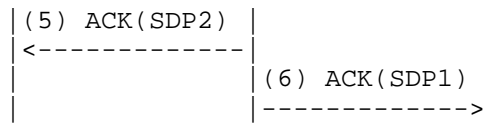


Figure 2: Role Conflict Flow

This flow is a variation on flow III of RFC 3725 [RFC3725]. In fact, it works better than flow III since it produces fewer messages. In this flow, the controller sends an offerless INVITE to agent A, which responds with its offer, SDP1. The agent then sends an offerless INVITE to agent B, which it responds to with its offer, SDP2. The controller then uses the offer from each agent to generate the answers. When this flow is used, ICE will run between agents A and B, but both will believe they are in the controlling role. With the role conflict resolution procedures, this flow will function properly when ICE is used.

At this time, there are no documented flows that can result in the case where both agents believe they are controlled. However, the conflict resolution procedures allow for this case, should a flow arise that would fit into this category.

#### Appendix D. Why Send an Updated Offer?

Section 11.1 describes rules for sending media. Both agents can send media once ICE checks complete, without waiting for an updated offer. Indeed, the only purpose of the updated offer is to "correct" the SDP so that the default destination for media matches where media is being sent based on ICE procedures (which will be the highest-priority nominated candidate pair).

This begs the question -- why is the updated offer/answer exchange needed at all? Indeed, in a pure offer/answer environment, it would not be. The offerer and answerer will agree on the candidates to use through ICE, and then can begin using them. As far as the agents themselves are concerned, the updated offer/answer provides no new information. However, in practice, numerous components along the signaling path look at the SDP information. These include entities performing off-path QoS reservations, NAT traversal components such as ALGs and Session Border Controllers (SBCs), and diagnostic tools that passively monitor the network. For these tools to continue to function without change, the core property of SDP -- that the existing, pre-ICE definitions of the addresses used for media -- the m and c lines and the rtcp attribute -- must be retained. For this reason, an updated offer must be sent.

Authors' Addresses

Marc Petit-Huguenin  
Impedance Mismatch

Email: [petithug@acm.org](mailto:petithug@acm.org)

Ari Keranen  
Ericsson  
Jorvas 02420  
Finland

Email: [ari.keranen@ericsson.com](mailto:ari.keranen@ericsson.com)

MMUSIC  
Internet-Draft  
Intended status: Standards Track  
Expires: April 7, 2013

T. Reddy  
P. Patil  
D. Wing  
Cisco  
October 4, 2012

Happy Eyeballs Extension for ICE  
draft-reddy-mmusic-ice-happy-eyeballs-00

Abstract

This document specifies requirements for algorithms that make ICE connectivity checks more aggressive to reduce delays in dual stack host connectivity checks when there is a path failure for the address family preferred by the application or by the operating system. As IPv6 is usually preferred, the procedures in this document helps avoid user-noticable 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 7, 2013.

Copyright Notice

Copyright (c) 2012 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. Candidates Priority . . . . .	3
4. Algorithm overview . . . . .	4
4.1. Processing the Results . . . . .	5
5. Relayed Candidates . . . . .	7
6. Setting Te, Tr and MAX_PAIRS_HAPPYEYE_STAGE . . . . .	8
7. IANA Considerations . . . . .	8
8. Security Considerations . . . . .	8
9. References . . . . .	8
9.1. Normative References . . . . .	8
9.2. Informative References . . . . .	9
Authors' Addresses . . . . .	9

## 1. Introduction

In situations where there are many IPv6 addresses, ICE [RFC5245] will prefer IPv6 [RFC6724] and will attempt connectivity checks on all the IPv6 candidates before trying an IPv4 candidate. If the IPv6 path is broken, this fallback to IPv4 can consume a lot of time, harming user satisfaction of dual stack devices.

This document describes an algorithm that makes ICE connectivity checks more responsive to failures of an address family by performing connectivity checks with both IPv6 and IPv4 candidates in parallel if IPv6 connectivity checks have not yet succeeded. This document specifies requirements for any such algorithm, with the goals that the ICE agent need not be inordinately harmed with a simple parallelisation of IPv6 and IPv4 connectivity checks and ensuring that the priority of precedence defined in [RFC6724] be honored.

For either of the address families, there is also a very realistic chance that connectivity checks for relayed candidates will always work. There are scenarios where firewalls block connectivity checks for Host/Server Reflexive candidates or for IPv4 or for IPv6. This document also proposes an optimization where connectivity checks with relayed checks are performed earlier than usual if connectivity checks using other candidates do not succeed.

## 2. Notational Conventions

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

This note uses terminology defined in [RFC5245].

## 3. Candidates Priority

A prioritization formula is used by ICE [RFC5245] so that most preferred address pairs are tested first, and if a sufficiently good pair is discovered, the tests can be stopped. With IPv6, addresses obtained from local network interfaces, called host candidates, are recommended as high-priority ones to be tested first since if they work, they provide usually the best path between the two hosts. The ICE specification recommends to use the rules defined in [RFC6724] as part of the prioritization formula for IPv6 host candidates and [I-D.keranen-mmusic-ice-address-selection] updates the ICE rules on how IPv6 host candidates are selected.



For dual stack hosts the preference for IPv6 host candidates is higher than IPv4 host candidates based on precedence value of IP addresses described in [RFC6724]. IPv6 server reflexive candidates have higher precedence than IPv4 server reflexive candidate since NPTv6 is stateless and transport-agnostic.

(highest)	IPv6 Host Candidate
	IPv4 Host Candidate
	IPv6 Server Reflexive Candidate
	IPv4 Server Reflexive Candidate
	IPv6 Relayed Transport Candidate
(lowest)	IPv4 Relayed Transport Candidate

Figure 1: Candidate Preferences in decreasing order

By using the technique in Section 4 IPv6 candidate pairs will be tested first as usual, but if connectivity checks are not successful after a certain period of time, the algorithm will become more aggressive and connectivity checks using IPv6/IPv4 host/server-reflexive candidates will be performed simultaneously. If connectivity checks with IPv6 candidate pairs do not yield any successful result then ICE endpoints can immediately start sending media using IPv4 host/server-reflexive candidates.

Note: [RFC6724] permits administrator to change the policy table to prefer IPv4 addresses over IPv6 addresses in which case the algorithm described in the next section is reversed.

#### 4. Algorithm overview

The Happy Eyeballs Extension for ICE is governed by a timer ( $T_e$ ) that is started just before carrying out the ICE connectivity checks for each check list under the following conditions:

1. when the candidates pairs include IPv6 and IPv4 addresses
2. list of IPv6 candidate pairs is higher than a configured threshold (`MAX_PAIRS_HAPPY_EYE_STAGE_I`). [RFC5245] recommends a limit of 100 for the candidate pairs.

When the timer ( $T_e$ ) fires, if the connectivity check using IPv6 candidate pairs are not yet successful and if the number of IPv6 candidate pairs with remote candidates of type host in the check list that are in Waiting and Frozen state are non-zero, the ICE agent performs the following Happy Eyeball steps in parallel with the regular ICE Ordinary checks:

- o Find the highest priority pair in the checklist that is in the Waiting state with candidate address family being IPv4 and remote candidate of type host. If there are no remote IPv6 candidates of type server-reflexive then IPv4 remote candidates of type server-reflexive will be added to the search.
  - 1. If there is such a pair then perform ICE connectivity check on this pair and set the state of the candidate pair to In-Progress.
  - 2. If there is no such pair find the highest priority pair in the checklist that is in the Frozen state with candidate address family being IPv4 and remote candidate of type host candidate. If there are no remote IPv6 candidates of type server-reflexive then IPv4 remote candidates of type server-reflexive will be added to the search. If there is such pair in Frozen state then unfreeze the pair, perform connectivity check on this pair and set the state of the candidate pair to In-Progress.
- o The above mentioned steps will be followed every  $T_a$  milliseconds and stopped when any of the below conditions are met:
  - 1. All IPv6 candidate pairs with remote candidates of type host in the check list are in any of the following states Succeeded, In-Progress or Failed states. The parallel activity is not required beyond this point because the regular ICE algorithm will itself pick up IPv4 candidate pairs not yet tested.
  - 2. All IPv4 candidate pairs with remote candidates of type host/server reflexive are in any of the following states Succeeded, In-Progress or Failed states.

#### 4.1. Processing the Results

If ICE connectivity checks using an IPv4 candidate is successful then ICE Agent will performs as usual "Discovering Peer Reflexive Candidates" (Section 7.1.3.2.1 of [RFC5245]), "Constructing a Valid Pair" (Section 7.1.3.2.2 of [RFC5245]), "Updating Pair States" (Section 7.1.3.2.3 of [RFC5245]), "Updating the Nominated Flag" (Section 7.1.3.2.4 of [RFC5245]).

If ICE connectivity checks using an IPv4 candidate is successful for each component of the media stream and connectivity checks using IPv6 candidates is not yet successful, the ICE endpoint will declare victory, conclude ICE for the media stream and start sending media using IPv4. However, it is also possible that ICE endpoint continues

to perform ICE connectivity checks with IPv6 candidate pairs and if checks using higher-priority IPv6 candidate pair is successful then media stream can be moved to the IPv6 candidate pair. Continuing to perform connectivity checks can be useful for subsequent connections, to optimize which connectivity checks are tried first. Such optimization is out of scope of this document.

The following diagram shows the behaviour during the connectivity check when Alice calls Bob and Agent Alice is the controlling agent and uses the aggressive nomination algorithm. "USE-CAND" implies the presence of the USE-CANDIDATE attribute.

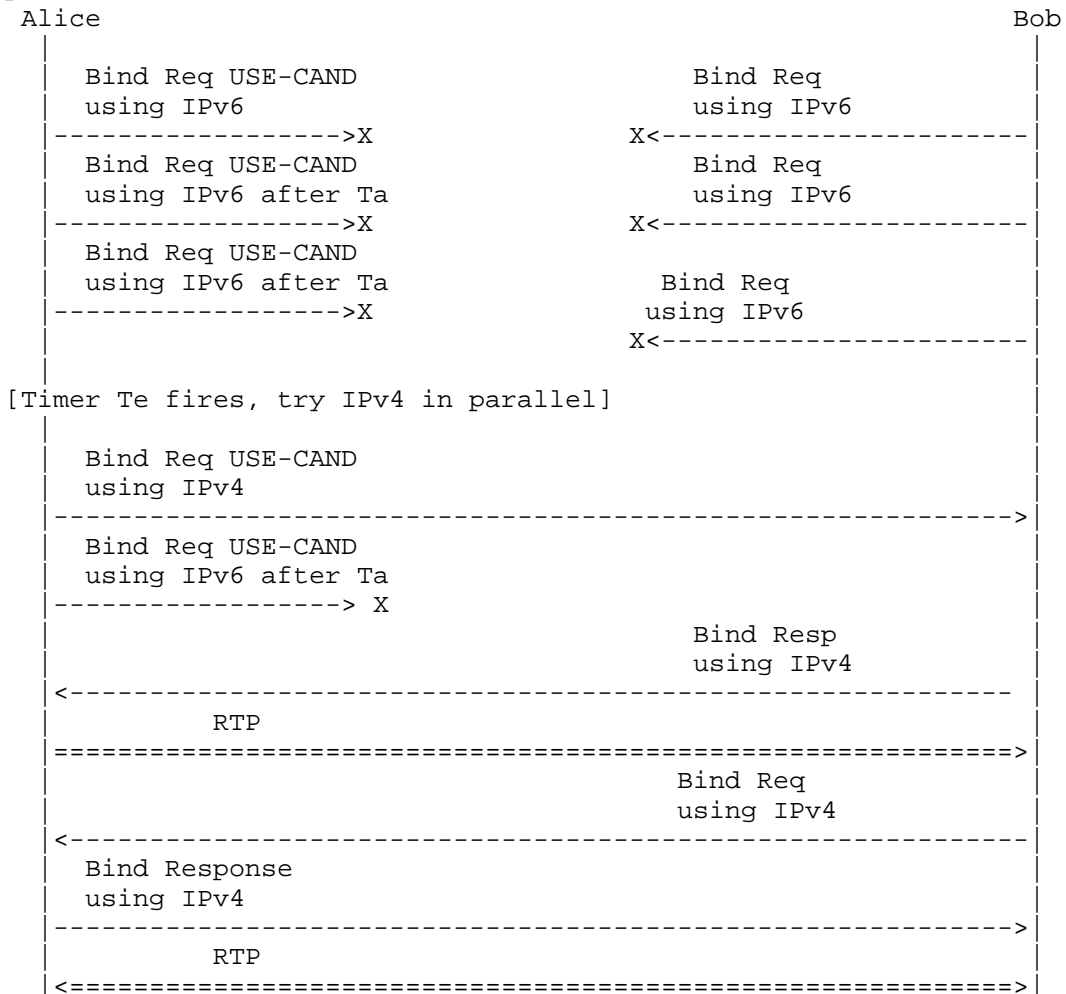


Figure 2: Happy Eyeballs Extension for ICE

## 5. Relayed Candidates

The optimization proposes doing connectivity checks with relayed candidates in parallel with other candidates. The algorithm does not make a distinction between IPv6/IPv4 relayed candidates and will choose the existing order among relayed candidate pair defined by ICE. If ICE connectivity check is successful using a relayed candidate from either of the IP address families, the ICE agent can stop connectivity checks for other relayed candidates.

This part of the Happy Eyeballs Extension for ICE is governed by a timer (Tr) that is started just before carrying out the ICE connectivity checks for each check list under the following conditions:

1. when the candidates pairs include IPv6 and IPv4 relayed addresses
2. list of candidate pairs is higher than a configured threshold (MAX\_PAIRS\_HAPPYEYE\_STAGE\_I).

When the timer (Tr) fires, If no ICE connectivity checks are successful as yet and if ICE Connectivity checks using IPv6 and IPv4 local relayed candidates have not yet been attempted then the following steps will be started by the ICE agent in parallel with other connectivity checks:

- o Find the highest priority pair in the checklist that is in the Waiting state with local candidate of type relayed.
1. If there is such a pair then perform ICE connectivity check on this pair and set the state of the candidate pair to In-Progress.
  2. If there is no such pair find the highest priority pair in the checklist that is in the Frozen state with local candidate of type relayed. If there is such pair in Frozen state then unfreeze the pair, perform connectivity check on this pair and set the state of the candidate pair to In-Progress.

If ICE connectivity checks using relayed candidate is successful then ICE Agent will performs as usual "Constructing a Valid Pair" (Section 7.1.3.2.2 of [RFC5245]), "Updating Pair States" (Section 7.1.3.2.3 of [RFC5245]), "Updating the Nominated Flag" (Section 7.1.3.2.4 of [RFC5245]). If ICE connectivity checks using local relayed candidates is successful for each component of the media stream and connectivity checks using higher priority candidate pairs has not yet succeeded then conclude ICE for the media stream and proceed to send media using local relayed candidate.

However ICE connectivity checks MUST be continued and if the check succeeds for a pair whose priority is higher than the previously selected candidate pair then media session will be moved to this pair. Hence media will only be sent briefly on TURN relays. Additional TURN server load is created due to this recommendations, especially when connectivity check using IPv6/IPv4 host/server-reflexive candidates are not completing quickly and the side affect could be that RTP receivers will receive packets out of order during switchover.

## 6. Setting Te, Tr and MAX\_PAIRS\_HAPPYEYE\_STAGE

The value of Ta, Tr, MAX\_PAIRS\_HAPPYEYE\_STAGE\_I, MAX\_PAIRS\_HAPPYEYE\_STAGE\_II and SHOULD be configurable, and SHOULD have a default of:

```
Te : 150ms
Tr : 500ms
MAX_PAIRS_HAPPYEYE_STAGE_I : 12
MAX_PAIRS_HAPPYEYE_STAGE_II : 6
```

Figure 3: Default Values

## 7. IANA Considerations

None.

## 8. 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.

## 9. References

### 9.1. Normative References

[I-D.keranen-mmusic-ice-address-selection]  
Keranen, A. and J. Arkko, "Update on Candidate Address Selection for Interactive Connectivity Establishment (ICE)", draft-keranen-mmusic-ice-address-selection-01 (work in progress), July 2012.

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

## 9.2. Informative References

- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

## Authors' Addresses

Tirumaleswar Reddy  
Cisco Systems, Inc.  
Cessna Business Park, Varthur Hobli  
Sarjapur Marathalli Outer Ring Road  
Bangalore, Karnataka 560103  
India

Email: tireddy@cisco.com

Prashanth Patil  
Cisco Systems, Inc.  
Cessna Business Park, Varthur Hobli  
Sarjapur Marthalli Outer Ring Road  
Bangalore, Karnataka 560103  
India

Email: praspatti@cisco.com

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

Email: [dwing@cisco.com](mailto:dwing@cisco.com)





MMUSIC  
Internet-Draft  
Intended status: Standards Track  
Expires: July 21, 2013

D. Wing  
P. Patil  
T. Reddy  
P. Martinsen  
Cisco  
January 17, 2013

Mobility with ICE (MICE)  
draft-wing-mmusic-ice-mobility-03

Abstract

This specification describes how endpoint mobility can be achieved using ICE. Two mechanisms are shown, one where both endpoints support ICE and another where only one endpoint supports ICE.

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 21, 2013.

Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	3
2. Notational Conventions . . . . .	4
3. Break Before Make . . . . .	4
3.1. Absence of other interfaces in Valid list . . . . .	5
3.1.1. Receiving ICE Mobility event . . . . .	6
3.2. Keeping unused relayed candidates active . . . . .	7
3.3. New STUN Attributes . . . . .	8
4. Make Before Break . . . . .	8
5. Mobility using TURN . . . . .	8
5.1. Creating an Allocation . . . . .	9
5.1.1. Sending an Allocate Request . . . . .	9
5.1.2. Receiving an Allocate Request . . . . .	10
5.1.3. Receiving an Allocate Success Response . . . . .	10
5.1.4. Receiving an Allocate Error Response . . . . .	10
5.2. Refreshing an Allocation . . . . .	11
5.2.1. Sending a Refresh Request . . . . .	11
5.2.2. Receiving a Refresh Request . . . . .	11
5.2.3. Receiving a Refresh Response . . . . .	11
5.3. New STUN Attribute MOBILITY-TICKET . . . . .	12
5.4. New STUN Error Response Code . . . . .	12
6. IANA Considerations . . . . .	12
7. Security Considerations . . . . .	12
7.1. Considerations for ICE mechanism . . . . .	12
7.2. Considerations for TURN mechanism . . . . .	13
8. Acknowledgements . . . . .	13
9. Change History . . . . .	13
9.1. Changes from draft-wing-mmusic-ice-mobility-00 to -01 . .	13
9.2. Changes from draft-wing-mmusic-ice-mobility-01 to -02 . .	13
9.3. Changes from draft-wing-mmusic-ice-mobility-02 to -03 . .	13
10. References . . . . .	13
10.1. Normative References . . . . .	13
10.2. Informative References . . . . .	14
Appendix A. . . . .	14
A.1. Presence of other interfaces in Valid list . . . . .	14
A.1.1. Receiving ICE Mobility event . . . . .	15
A.2. Losing an Interface . . . . .	15
A.2.1. Keeping unused candidates in the valid list active . .	16
Authors' Addresses . . . . .	16

## 1. Introduction

When moving between networks, an endpoint has to change its IP address. This change breaks upper layer protocols such as TCP and RTP. Various techniques exist to prevent this breakage, all tied to making the endpoint's IP address static (e.g., Mobile IP, Proxy Mobile IP, LISP). Other techniques exist, which make the upper layer protocol ambivalent to IP address changes (e.g., SCTP). The mechanisms described in this document are in that last category.

ICE [RFC5245] ensures two endpoints have a working media path between them, and is typically used by Internet-connected interactive media systems (e.g., SIP endpoints). ICE does not expect either the local host or the remote host to change their IP addresses. Although ICE does allow an "ICE restart", this is done by sending a re-INVITE which goes over the SIP signaling path. The SIP signaling path is often slower than the media path (which needs to be recovered as quickly as possible), consumes an extra half round trip, and incurs an additional delay if the mobility event forces the endpoint to re-connect with its SIP proxy. When a device changes its IP address, it is necessary for it to re-establish connectivity with its SIP proxy, which can be performed in parallel with the steps described in this document. This document describes how mobility is performed entirely in the media path, without the additional delay of re-establishing SIP connectivity, issuing a new offer/answer, or the complications of multiple SIP offers. This document considers re-establishing bi-directional media the most critical aspect of a successful mobility event, and its efforts are towards meeting that goal.

A TURN [RFC5766] server relays media packets and is used for a variety of purposes, including overcoming NAT and firewall traversal issues and IP address privacy. The existing TURN specification does not allow the client address to change, especially if multiple clients share the same TURN username (e.g., the same credentials are used on multiple devices).

This document proposes two mechanisms to achieve RTP mobility: a mechanism where both endpoints support ICE, and a mechanism where only one endpoint supports ICE. When both endpoints support ICE, ICE itself can be used to provide mobility. When only one endpoint supports ICE, a TURN server provides mobility. Both mobility techniques work across and between network types (e.g., between 3G and wired Internet access), so long as the client can still access the remote ICE peer or TURN server.

Readers are assumed to be familiar with ICE [RFC5245].

## 2. Notational Conventions

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

This note uses terminology defined in [RFC5245], and the following additional terminology:

**Break Before Make:** The initially selected interface for communication may become unavailable (e.g due to loss of coverage when moving out of a WiFi hotspot) and new interfaces may become available due to administrative action (e.g manual activation of a specific connectivity technology) or due to dynamic conditions (e.g. Entering coverage area of a wireless network).

**Make Before Break:** The initially selected interface for communication may become deprioritized (e.g new interface becoming available and it's per bit cost is cheaper and the connection speed is faster than existing interface used for communication).

**Simultaneous Mobility:** If both the endpoints are mobile and roam at the same time between networks.

## 3. Break Before Make

When both endpoints support ICE, ICE itself can provide mobility functions. One of the primary aspects of ICE is its address gathering, wherein ICE has each endpoint determine all of the IP addresses and ports that might be usable for that endpoint and communicate that list of addresses and ports to its peer, usually over SDP. That enables the next primary aspect of ICE, which is its connectivity checks: each ICE endpoint sends a connectivity check to that list of addresses and ports. A connectivity check may unknowingly traverse a NAT, which means the ICE endpoint receiving the connectivity check cannot validate the source IP address or port of the connectivity against the list of IP addresses and ports provided by the ICE peer. In fact, if the source IP address and port is not known to the ICE endpoint, it is added to the list of candidates (Section 7.2.1.3 of [RFC5245]). ICE Mobility takes advantage of that existent ICE functionality.

Endpoints that support ICE Mobility perform ICE normally, and MUST also include the MOBILITY-SUPPORT attribute in all of their STUN requests and their STUN responses. The inclusion of this attribute allows the ICE peer to determine if it can achieve mobility using ICE or needs to use TURN. To force the use of TURN to achieve ICE

mobility, the ICE endpoint SHOULD NOT respond to ICE connectivity checks that have an IP address and port different from the TURN server, unless those connectivity checks contain the MOBILITY-SUPPORT attribute. In this way, the remote peer will think those other candidates are invalid (because its connectivity checks did not succeed).

After concluding ICE and moving to the ICE completed state (see Section 8 of [RFC5245] either endpoint or both endpoints can initiate ICE Mobility, no matter if it was the Controlling Agent or the Controlled Agent during normal ICE processing.

### 3.1. Absence of other interfaces in Valid list

When the interface currently being used for communication becomes unavailable then ICE agent acquires a list of interfaces that are available and based on the locally configured host policy preferences, the ICE endpoint performs ICE Mobility using one of the available interfaces. In this case local candidates from the selected interface are not present in the valid list. ICE Mobility is performed by :

1. The ICE agent remembers the remote host/server-reflexive candidates for each component of the media streams previously used from the valid list before clearing its ICE check list and ICE Valid List.
2. The ICE endpoint gathers host candidates on the new interface, forms a check list by creating candidate pairs with local host candidates and remote host/server-reflexive candidates collected in step 1, performs "Computing Pair Priority and Ordering Pairs" (Section 5.7.2 of [RFC5245]), "Pruning the Pairs" (Section 5.7.3 of [RFC5245]), "Computing states" (Section 5.7.4 of [RFC5245]).
3. The ICE endpoint initiates ICE connectivity checks on those candidates from the check list in the previous step, and includes the MOBILITY-EVENT attribute in those connectivity checks.
4. The ICE endpoint acts as controlling agent and the ICE connectivity check from the previous step SHOULD also include the USE-CANDIDATE attribute to signal an aggressive nomination (see Section 2.6 of [RFC5245]). An aggressive nomination allows sending media immediately after the connectivity check completes, without waiting for other connectivity checks to complete.
5. The ICE endpoint performs "Discovering Peer Reflexive Candidates" (Section 7.1.3.2.1 of [RFC5245]), "Constructing a Valid Pair" (Section 7.1.3.2.2 of [RFC5245]), "Updating Pair States" (Section

7.1.3.2.3 of [RFC5245]), and "Updating the Nominated Flag" (Section 7.1.3.2.4 of [RFC5245]). When the valid list contains a candidate pair for each component then ICE processing is considered complete for the media stream and ICE agent can start sending media using highest-priority nominated candidate pair.

6. Once ICE connectivity checks for all of the media streams are completed, the controlling ICE endpoint follows the procedures in Section 11.1 of [RFC5245], specifically to send updated offer if the candidates in the m and c lines for the media stream (called the DEFAULT CANDIDATES) do not match ICE's SELECTED CANDIDATES (also see Appendix B.9 of [RFC5245]).

The ICE endpoint even after Mobility using ICE is successful can issue an updated offer indicating ICE restart if connectivity checks using higher priority candidate pairs are not successful.

Mobility using ICE could fail in case of Simultaneous Mobility or if the ICE peer is behind NAT that performs Address-Dependent Filtering (see Section 5 of [RFC5245]). Hence the ICE endpoint in parallel will re-establish connection with the SIP proxy. It will then determine whether to initiate ICE restart under the following conditions :

1. After re-establishing connection with the SIP proxy and before sending new offer to initiate ICE restart if Mobility using ICE is successful then stop sending the new offer.
2. After successful negotiation of updated offer/answer to initiate ICE restart, proceed with ICE restart and stop Mobility using ICE if ICE checks are in the Running/Failed states or ICE is partially successful and not yet reached ICE complete state. It's not implementation friendly to have to two checks running in parallel. ICE restart can re-use partial successful ICE connectivity check results from Mobility using ICE if required as optimization.

#### 3.1.1.1. Receiving ICE Mobility event

A STUN Binding Request containing the MOBILITY-EVENT attribute MAY be received by an ICE endpoint. The agent MUST use short-term credential to authenticate the STUN request containing the MOBILITY-EVENT attribute and perform a message integrity check. The ICE endpoint will generate STUN Binding Response containing the MOBILE-SUPPORT attribute and the ICE agent takes role of controlled agent. If STUN Request containing the MOBILITY-EVENT attribute is received before the endpoint is in the ICE Completed state, it should be silently discarded.

The agent remembers the highest-priority nominated pairs in the Valid list for each component of the media stream, called the previous selected pairs before removing all the selected candidate pairs from the Valid List . It continues sending media to that address until it finishes with the steps described below. Because those packets might not be received due to the mobility event, it MAY cache a copy of those packets.

1. The ICE endpoint constructs a pair whose local candidate is equal to the transport address on which the STUN request was received with MOBILITY-EVENT, USE-CANDIDATE attributes and a remote candidate equal to the source transport address where the STUN request came from.
2. The ICE endpoint will add this pair to the valid list if not already present.
3. The agent sets the nominated flag for that pair in the valid pair to true. ICE processing is considered complete for a media stream if the valid list contains a selected candidate pair for each component and ICE agent can start sending media.

The ICE endpoint will follow Steps 1 to 3 when subsequent STUN Binding Requests are received with MOBILITY-EVENT and USE-CANDIDATE attributes.

### 3.2. Keeping unused relayed candidates active

The ICE endpoints can maintain the relayed candidates active even when not actively used, so that relayed candidates can be tried if ICE connectivity checks using other candidate types fails. The ICE agent will have to create permissions in the TURN server for the remote relayed candidate IP addresses and perform the following steps :

1. The ICE agent will keep the relayed candidates alive using Refresh transaction, as described in [RFC5766].
2. When the endpoint IP address changes due to mobility, the ICE agent will refresh it's allocation with TURN server using Section 5.2.
3. The ICE agent will pair local and remote relayed candidates for connectivity checks when performing the steps in Section 3.1.
4. If the ICE connectivity check succeeds only with local and remote relayed candidates, it suggests that either other peer is roaming at the same time or is behind Address-Dependent Filtering NAT.

The ICE agent adds the relayed candidate pair to the valid list and marks it as selected. The ICE agent can now send media using the newly selected relayed candidate pair. The Mobile device must re-establish connection with SIP proxy, issue an updated offer indicating ICE restart so that media can be switched to higher-priority candidate pairs.

This approach assists Mobility using ICE to succeed but brings in additional overhead of maintaining relayed candidates. In case of Simultaneous Mobility, host candidates can change for both the endpoints by maintaining relayed candidates and using Section 5 media session can be established using the relayed candidate pair.

### 3.3. New STUN Attributes

Three new attributes are defined by this section: MOBILITY-EVENT, MOBILITY-SUPPORT.

The MOBILITY-EVENT attribute indicates the sender experienced a mobility event. This attribute has no value, thus the attribute length field MUST always be 0. Rules for sending and interpretation of receiving are described above.

The MOBILITY-SUPPORT attribute indicates the sender supports ICE Mobility, as defined in this document. This attribute has no value, thus the attribute length field MUST always be 0. Rules for sending and interpretation of receiving are described above.

## 4. Make Before Break

When a new interface comes up and initially selected interface becomes deprioritized (e.g. due to a low cost interface becoming available). The ICE endpoint re-connects to the SIP proxy using the new interface, gathers candidates, exchanges updated offer/exchange to restart ICE. Once ICE processing has reached the Completed state then the ICE endpoint can successfully switch the media over to the new interface. The interface initially used for communication can now be turned off without disrupting communications.

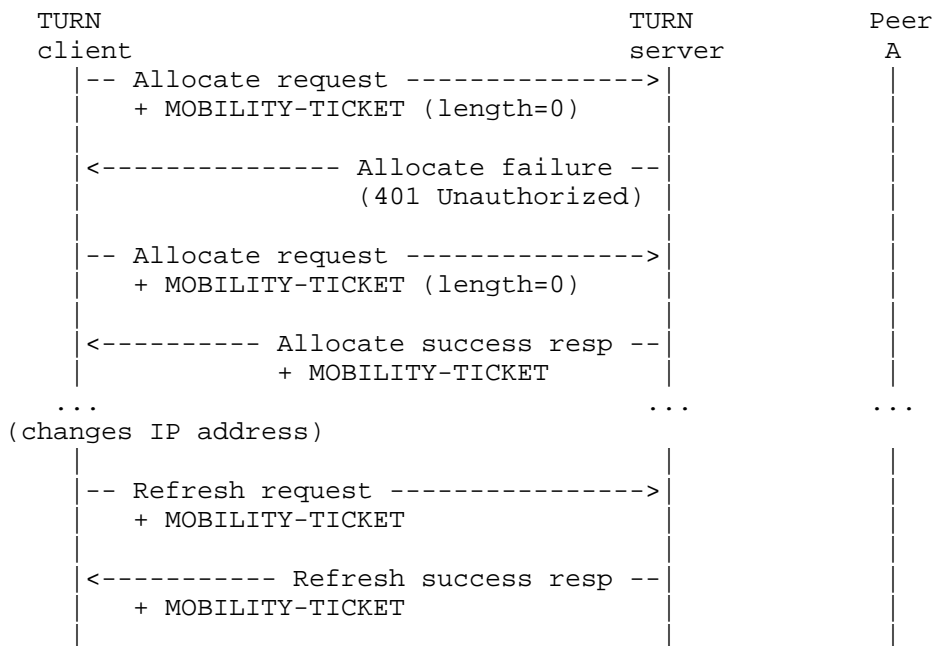
## 5. Mobility using TURN

To achieve mobility, a TURN client should be able to retain an allocation on the TURN server across changes in the client IP address as a consequence of movement to other networks.

When the client sends the initial Allocate request to the TURN



server, it will also include the new STUN attribute MOBILITY-TICKET (with zero length value), which indicates that the client is capable of mobility and desires a ticket. The TURN server provisions a ticket that is sent inside the new STUN attribute MOBILITY-TICKET in the Allocate Success response to the client. The ticket will be used by the client when it wants to refresh the allocation but with a new client IP address and port. It also ensures that the allocation can only be refreshed this way by the same client. When a client's IP address changes due to mobility, it presents the previously obtained ticket in a Refresh Request to the TURN server. If the ticket is found to be valid, the TURN server will retain the same relayed address/port for the new IP address/port allowing the client to continue using previous channel bindings -- thus, the TURN client does not need to obtain new channel bindings. Any data from external peer will be delivered by the TURN server to this new IP address/port of the client. The TURN client will continue to send application data to its peers using the previously allocated channelBind Requests.



## 5.1. Creating an Allocation

### 5.1.1. Sending an Allocate Request

In addition to the process described in Section 6.1 of [RFC5766], the client includes the MOBILITY-TICKET attribute with length 0. This

indicates the client is a mobile node and wants a ticket.

#### 5.1.2. Receiving an Allocate Request

In addition to the process described in Section 6.2 of [RFC5766], the server does the following:

If the MOBILITY-TICKET attribute is included, and has length zero, and the TURN session mobility is forbidden by local policy, the server MUST reject the request with the new Mobility Forbidden error code. Following the rules specified in [RFC5389], if the server does not understand the MOBILITY-TICKET attribute, it ignores the attribute.

If the server can successfully process the request create an allocation, the server replies with a success response that includes a STUN MOBILITY-TICKET attribute. TURN server stores it's session state, such as 5-tuple and NONCE, into a ticket that is encrypted by a key known only to the TURN server and sends the ticket in the STUN MOBILITY-TICKET attribute as part of Allocate success response.

The ticket is opaque to the client, so the structure is not subject to interoperability concerns, and implementations may diverge from this format. TURN Allocation state information is encrypted using 128-bit key for Advance Encryption Standard (AES) and 256-bit key for HMAC-SHA-256 for integrity protection.

#### 5.1.3. Receiving an Allocate Success Response

In addition to the process described in Section 6.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the response. This attribute will be presented by the client to the server during a subsequent Refresh request to aid mobility.

#### 5.1.4. Receiving an Allocate Error Response

If the client receives an Allocate error response with error code TBD (Mobility Forbidden), the error is processed as follows:

- o TBD (Mobility Forbidden): The request is valid, but the server is refusing to perform it, likely due to administrative restrictions. The client considers the current transaction as having failed. The client MAY notify the user or operator and SHOULD NOT retry the same request with this server until it believes the problem has been fixed.

All other error responses must be handled as described in [RFC5766].

## 5.2. Refreshing an Allocation

### 5.2.1. Sending a Refresh Request

If a client wants to refresh an existing allocation and update its time-to-expiry or delete an existing allocation, it will send a Refresh Request as described in Section 7.1 of [RFC5766]. If the client wants to retain the existing allocation in case of IP change, it will include the MOBILITY-TICKET attribute received in the Allocate Success response. If a Refresh transaction was previously made, the MOBILITY-TICKET attribute received in the Refresh Success response of the transaction must be used.

### 5.2.2. Receiving a Refresh Request

In addition to the process described in Section 7.2 of [RFC5766], the client does the following:

If the STUN MOBILITY-TICKET attribute is included in the Refresh Request then the server will not retrieve the 5-tuple from the packet to identify an associated allocation. Instead TURN server will decrypt the received ticket, verify the ticket's validity and retrieve the 5-tuple allocation from the contents of the ticket. If this 5-tuple obtained from the ticket does not identify an existing allocation then the server MUST reject the request with an error.

If the source IP address and port of the Refresh Request is different from the stored 5-tuple allocation, the TURN server proceeds with checks to see if NONCE in the Refresh request is the same as the one provided in the ticket. The TURN server also uses MESSAGE-INTEGRITY validation to identify that it is the same user which had previously created the TURN allocation. If the above checks are not successful then server MUST reject the request with a 441 (Wrong Credentials) error.

If all of the above checks pass, the TURN server understands that the client has moved to a new network and acquired a new IP address. The source IP address of the request could either be the host transport address or server-reflexive transport address. The server then updates its 5-tuple with the new client IP address and port. TURN server calculates the ticket with the new 5-tuple and sends the new ticket in the STUN MOBILITY-TICKET attribute as part of Refresh Success response.

### 5.2.3. Receiving a Refresh Response

In addition to the process described in Section 7.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the

response. This attribute will be presented by the client to the server during a subsequent Refresh Request to aid mobility.

### 5.3. New STUN Attribute MOBILITY-TICKET

This attribute is used to retain an Allocation on the TURN server. It is exchanged between the client and server to aid mobility. The value is encrypted and identifies session state such as 5-tuple and NONCE. The value of MOBILITY-TICKET is a variable-length value.

### 5.4. New STUN Error Response Code

This document defines the following new error response code:

Mobility Forbidden: Mobility request was valid but cannot be performed due to administrative or similar restrictions.

## 6. IANA Considerations

IANA is requested to add the following attributes to the STUN attribute registry [iana-stun],

- o MOBILITY-TICKET (0x802E, in the comprehension-optional range)
- o MOBILITY-EVENT (0x802, in the comprehension-required range)
- o MOBILITY-SUPPORT (0x8000, in the comprehension-optional range)

and to add a new STUN error code "Mobility Forbidden" with the value 501 to the STUN Error Codes registry [iana-stun].

## 7. Security Considerations

### 7.1. Considerations for ICE mechanism

A mobility event only occurs after both ICE endpoints have exchanged their ICE information. Thus, both username fragments are already known to both endpoints. Each endpoint contributes at least 24 bits of randomness to the ice-ufrag (Section 15.4 of [RFC5245]), which provides 48 bits of randomness. An off-path attacker would have to guess those 48 bits to cause the endpoints to perform HMAC-SHA1 validation of the MESSAGE-INTEGRITY attribute.

An attacker on the path between the ICE endpoints will see both ice-ufrags, and can cause the endpoints to perform HMAC-SHA1 validation

by sending messages from any IP address.

## 7.2. Considerations for TURN mechanism

TURN server MUST use strong encryption and integrity protection for the ticket to prevent an attacker from using a brute force mechanism to obtain the ticket's contents or refreshing allocations.

Security considerations described in [RFC5766] are also applicable to this mechanism.

## 8. Acknowledgements

Thanks to Alfred Heggstad, Lishitao, Sujing Zhou, Martin Thomson, Emil Ivov for review and comments.

## 9. Change History

[Note to RFC Editor: Please remove this section prior to publication.]

### 9.1. Changes from draft-wing-mmusic-ice-mobility-00 to -01

- o Updated section 3

### 9.2. Changes from draft-wing-mmusic-ice-mobility-01 to -02

- o Updated Introduction, Notational Conventions, sections 3.1, 3.2.
- o Updated section 3.5

### 9.3. Changes from draft-wing-mmusic-ice-mobility-02 to -03

- o Moved sections Presence of other interfaces in Valid list, Losing an Interface to Appendix.

## 10. References

### 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT)

Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

## 10.2. Informative References

[RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, May 2010.

[RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.

[iana-stun] IANA, "IANA: STUN Attributes", April 2011, <<http://www.iana.org/assignments/stun-parameters/stun-parameters.xml>>.

## Appendix A.

### A.1. Presence of other interfaces in Valid list

This technique is optional and only relevant if there is a host policy to maintain unused candidates on other interfaces using the steps in Appendix A.2.1. ICE Agent can maintain unused candidates on other interfaces if it detects that it is behind Address-Dependent Filtering NAT or Firewall. ICE Agent can detect NAT, Firewall behaviour using the procedure explained in [RFC5780]. When the interface currently being used for media communication becomes unavailable. If other interfaces are available and local candidates from these interfaces are already present in the valid list then ICE endpoint will perform the following steps :

1. The ICE endpoint based on the locally configured host policy preferences, will select a interface whose candidates are already present in the valid list.
2. The ICE endpoint clears all the pairs in the valid list containing the IP addresses from the interface that become

unavailable.

3. The ICE endpoint initiates ICE connectivity checks on the selected interface. The ICE endpoint acts as controlling agent and MUST include MOBILITY-EVENT attribute to signal mobility event and SHOULD also include the USE-CANDIDATE attribute to signal an aggressive nomination (see Section 2.6 of [RFC5245]). When all components have a nominated pair in the valid list, media can begin to flow using the highest priority nominated pair.
4. The ICE endpoint will re-establish connection with the SIP proxy. Once ICE connectivity checks for all of the media streams are completed, the controlling ICE endpoint follows the procedures in Section 11.1 of [RFC5245], specifically to send updated offer if the candidates in the m and c lines for the media stream (called the DEFAULT CANDIDATES) do not match ICE's SELECTED CANDIDATES (also see Appendix B.9 of [RFC5245]).

The ICE endpoint after Mobility using ICE is successful can issue an updated offer indicating ICE restart if higher priority interface becomes available.

#### A.1.1.1. Receiving ICE Mobility event

The ICE endpoint that receives ICE Mobility Event will perform the steps in Section 3.1.1.

#### A.2. Losing an Interface

When an interface is lost, the SDP MAY be updated, so that the remote ICE host does not waste its efforts with connectivity checks to that address, as those checks will fail. Because it can be argued that this is merely an optimization, and that the interface loss might be temporary (and soon regained), and that ICE has reasonable accommodation for candidates where connectivity checks timeout, this specification does not strongly encourage updating the SDP to remove a lost interface.

Likewise, this specification recommends that ICE candidate addresses in valid list be maintained actively, subject to the host's policy. For example, battery operated hosts have a strong incentive to not maintain NAT binding for server reflexive candidates learnt through STUN Binding Request, as the maintenance requires sending periodic STUN Binding Indication. As another example, a host that is receiving media over IPv6 may not want to persist with keeping a NATted IPv4 mapping alive (because that consumes a NAT mapping that could be more useful to a host actively utilizing the mapping for

real traffic).

Note: this differs from Section 8.3 of [RFC5245], which encourages abandoning unused candidates.

#### A.2.1. Keeping unused candidates in the valid list active

ICE endpoint subject to host policy can continue performing ICE connectivity checks using candidates from other interfaces on the host even after ICE is complete. If valid list contains unused candidate pairs from other interfaces and one of these interfaces can be selected to send to media in case the existing interface used for media is unavailable then ICE endpoint can keep the unused candidate pairs from other interface{s} alive by sending keepalives every NN seconds. It is recommended to only keep host/server-reflexive candidates active in the valid list and not the relayed candidates.

##### A.2.1.1. Sending keep alive requests

Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows [RFC6263] describes various reasons for doing keepalives on inactive streams and how to keep NAT mapping alive. However this specification requires some additional functionality associated with the keepalives.

STUN binding requests MUST be used as the keepalive message instead of the STUN Binding indication as specified in [RFC5245]. This is to ensure positive peer consent from the remote side that the candidate pair is still active and in future mobility can be achieved using the steps in Appendix A.1 . The request must include the MOBILITY-SUPPORT attribute. If the STUN binding response matches a pair in the checklist then that candidate pair should be kept in the list. If the STUN transaction fails then the candidate pair will be removed from valid list.

##### A.2.1.2. Receiving keep alive requests

Upon receiving a STUN binding request containing a MOBILITY-SUPPORT attribute even when ICE processing is in the Completed state, the ICE endpoint will add this pair to the valid list if not already present and generate STUN Binding Response containing the MOBILE-SUPPORT attribute.



Authors' Addresses

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

Email: [dwing@cisco.com](mailto:dwing@cisco.com)

Prashanth Patil  
Cisco Systems, Inc.  
Cessna Business Park, Varthur Hobli  
Sarjapur Marthalli Outer Ring Road  
Bangalore, Karnataka 560103  
India

Email: [praspati@cisco.com](mailto:praspati@cisco.com)

Tirumaleswar Reddy  
Cisco Systems, Inc.  
Cessna Business Park, Varthur Hobli  
Sarjapur Marathalli Outer Ring Road  
Bangalore, Karnataka 560103  
India

Email: [tiredy@cisco.com](mailto:tiredy@cisco.com)

Paal-Erik Martinsen  
Cisco Systems, Inc.  
Philip Pedersens vei 22  
Lysaker, Akershus 1325  
Norway

Email: [palmarti@cisco.com](mailto:palmarti@cisco.com)

