MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

K. Drage, Ed.
M. Makaraju
J. Stoetzer-Bradler
Alcatel-Lucent
R. Ejzak
J. Marcon
Unaffiliated

March 9, 2015

SDP-based Data Channel Negotiation
draft-ietf-mmusic-data-channel-sdpneg-01

Abstract

The Real-Time Communication in WEB-browsers (RTCWeb) working group is charged to provide protocols to support direct interactive rich communications using audio, video, and data between two peers' web-browsers. For the support of data communication, the RTCWeb working group has in particular defined the concept of bi-directional data channels over SCTP, where each data channel might be used to transport other protocols, called sub-protocols. Data channel setup can be done using either the internal in-band band (also referred to as 'internal' for the rest of the document) Data Channel Establishment Protocol or some external out-of-band simply referred to as 'external negotiation' in the rest of the document . This document specifies how the SDP offer/answer exchange can be used to achieve such an external negotiation. Even though data channels are designed for RTCWeb use initially they may be used by other protocols like, but not limited to, the CLUE protocol. This document is intended to be used wherever data channels are used.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Table of Contents

1. Introduction

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

This document defines SDP-based out-of-band negotiation procedures to establish data channels for transport of well-defined sub-protocols.

2. Conventions

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

3. Terminology

This document uses the following terms:

Data channel: A WebRTC data channel as specified in [I-D.ietf-rtcweb-data-channel].

Data channel stack: An entity which, upon application request, runs data channel protocol to keep track of states, sending and

receive data. If the application is browser based JavaScript application then this stack resides in the browser. If the application is a native application then this stack resides in application and accessible to it via some sort of APIs.

Data channel properties: fixed properties assigned to a data channel at the time of its creation. Some of these properties determine the way the data channel stack transmits data on this channel (e.g., stream identifier, reliability, order of delivery...).

DCEP - Data Channel Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol].

External negotiation: Data channel negotiation based on SDP offer/ answer outlined in this specification.

Internal negotiation: Data channel negotiation based on the Data Channel Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol].

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

In-band negotiation: data channel negotiation based on the Data Channel Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol].

Out-of-band: transmission through the application signaling path.

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

Stream identifier: the identifier of the outbound and inbound SCTP streams composing a data channel.

4. Data Channels

This section summarizes how data channels work in general. Note that the references to ’browser’ here is intentional as in this specific example the data channel user is a Webrtc enabled browser.

A WebRTC application creates a data channel via the data channel API, by providing a number of setup parameters (sub-protocol, label, reliability, order of delivery, priority). The application also specifies if it wants to make use of the in-band negotiation using the DCEP [I-D.ietf-rtcweb-data-protocol], or if the application

intends to perform an "external negotiation" using some other in-band or out-of-band mechanism.

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

```
m=application 54111 UDP/DTLS/SCTP webrtc-datachannel
c=IN IP4 79.97.215.79
a=max-message-size:100000
a=sctp-port 5000
a=setup:actpass
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
```

Note: A WebRTC browser will only use "m" line format "webrtc-datachannel", and will not use other formats in the "m" line for other protocols such as t38. [I-D.ietf-mmusic-sctp-sdp] supports only one SCTP association to be established on top of a DTLS session.

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

4.1. Stream Identifier Numbering

Independently from the requested type of negotiation, the application creating a data channel can either pass to the browser the stream identifier to assign to the data channel or else let the browser pick one identifier from the ones unused.

To avoid glare situations, each endpoint can moreover own an exclusive set of stream identifiers, in which case an endpoint can only create a data channel with a stream identifier it owns.

Which set of stream identifiers is owned by which endpoint is determined by convention or other means.

For data channels negotiated in-band, one endpoint owns by convention the even stream identifiers, whereas the other owns the odd stream identifiers, as defined in [I-D.ietf-rtcweb-data-protocol].

For data channels externally negotiated, no convention is defined by default.

4.2. Generic External Negotiation

4.2.1. Overview

In-band negotiation only provides for negotiation of data channel transport parameters and does not provide for negotiation of sub-protocol specific parameters. External negotiation can be defined to allow negotiation of parameters beyond those handled by in-band negotiation, e.g., parameters specific to the sub-protocol instantiated on a particular data channel. See Section 5.1.2 for an example of such a parameter.

The following procedures are common to all methods of external negotiation, whether in-band (communicated using proprietary means on an already established data channel) or out-of-band (using SDP or some other protocol associated with the signaling channel).

4.2.2. Opening a Data Channel

In the case of external negotiation, the endpoint application has the option to fully control the stream identifier assignments. However these assignments have to coexist with the assignments controlled by the data channel stack for the in-band negotiated data channels (if any). It is the responsibility of the application to ensure consistent assignment of stream identifiers.

When the application requests the creation of a new data channel to be set up via external negotiation, the data channel stack creates the data channel locally without sending any DATA_CHANNEL_OPEN message in-band, and sets the data channel state to Connecting if the SCTP association is not yet established, or sets the data channel state to Open if the SCTP association is already established. The side which starts external negotiation creates data channel using underlying data channel stack API and the data channel is put into open state immediately (assuming ICE, SCTP procedures were already done). However, the application can’t send data on this data channel until external negotiation is complete with the peer. This is because peer needs to be aware and accept the data channel via external negotiation. The peer after accepting the data channel offer can start sending data immediately. This implies that the offerer may get data channel message before external negotiation is complete and the application should be ready to handle it.

If the peer rejects the data channel part of the offer then it doesn’t have to do anything as the data channel was not created using the stack. The offerer on the other hand needs to close the data channel that was opened by invoking relevant data channel stack API procedures.

It is also worth noting that a data channel stack implementation may not provide any API to create and close data channels; instead the data channels are used on the fly as needed just by communicating via external means or by even having some local configuration/assumptions on both the peers.

The application then externally negotiates the data channel properties and sub-protocol properties with the peer's application.

[ASSUMPTION] The peer must then symmetrically create a data channel with these negotiated data channel properties. This is the only way for the peer's data channel stack to know which properties to apply when transmitting data on this channel. The data channel stack must allow data channel creation with any non-conflicting stream identifier so that both peers can create the data channel with the same stream identifier.

In case the external negotiation is correlated with an SDP offer/answer exchange that establishes the SCTP association, the SCTP initialization completion triggers a callback from the data channel stack to an application on both the ends to change the data channel state from Connecting to Open. The details of this interface is specific to the data channel user application. Browser based applications (could include hybrid apps) will use [WebRtcAPI], while native applications use a compatible API, which is yet to be specified. See Section 5.2.3 for details on when the data channel stack can assume the data channel is open, and on when the application can assume the data channel is open.

4.2.3. Closing a Data Channel

When the application requests the closing of an externally negotiated data channel, the data channel stack always performs an in-band SSN reset for this channel.

Depending upon the method used for external negotiation and the sub-protocol associated with the data channel, the closing might in addition be signaled to the peer via external negotiation.

5. SDP-based External Negotiation

This section defines a method of external negotiation by which two clients can negotiate data channel-specific and sub-protocol-specific parameters, using the out-of-band SDP offer/answer exchange. This SDP extension can only be used with SDP offer/answer model.

5.1. SDP Syntax

Two new SDP attributes are defined to support external negotiation of data channels. The first attribute provides for negotiation of channel-specific parameters. The second attribute provides for negotiation of sub-protocol-specific parameters.

5.1.1. SDP Attribute for Data Channel Parameter Negotiation

Associated with the SDP "m" line that defines the SCTP association for data channels (defined in Section 4), each SDP offer and answer includes one "a=dcmap:" attribute that defines the data channel parameters for each data channel to be negotiated. Each such attribute line specifies the following parameters for a data channel: SCTP stream identifier, sub-protocol, label, reliability, order of delivery, and priority.

The intention of exchanging these attributes is to create data channels on both the peers with the same set of attributes without actually using [I-D.ietf-rtcweb-data-protocol]. It is assumed that the data channel properties (reliable/partially reliable, ordered/ unordered) are suitable per the sub-protocol transport requirements.

5.1.1.1. dcmap Attribute

"a=dcmap:" is a media level attribute having following ABNF syntax.

Formal Syntax:

Name: dcmap

Value: dcmap-value

Usage Level: media

Charset Dependent: no

Syntax:

```
dcmap-value     = dcmap-stream-id
                  [ SP dcmap-opt *(";" dcmap-opt) ]
dcmap-opt       = ordering-opt / subprotocol-opt / label-opt
                  / maxretr-opt / maxtime-opt
                  ; Either only maxretr-opt or maxtime-opt
                  ; is present.
                  ; Both MUST not be present.

dcmap-stream-id = 1*DIGIT
ordering-opt    = "ordered=" ordering-value
ordering-value  = "true" / "false"
subprotocol-opt = "subprotocol=" quoted-string
label-opt       = "label=" quoted-string
maxretr-opt     = "max-retr=" maxretr-value
maxretr-value   = <from-Reliability-Parameter of
                   I-D.ietf-rtcweb-data-protocol>
                  ; number of retransmissions
maxtime-opt     = "max-time=" maxtime-value
maxtime-value   = <from-Reliability-Parameter of
                  I-D.ietf-rtcweb-data-protocol>
                  ; milliseconds

quoted-string   = DQUOTE *(quoted-char / escaped-char) DQUOTE
quoted-char     = SP / quoted-visible
quoted-visible  = %21 / %23-24 / %26-7E ; VCHAR without " or %
escaped-char    = "%" HEXDIG HEXDIG
DQUOTE          = <from-RFC5234>
integer         = <from-RFC5234>
```

Examples:

```
a=dcmap:0
a=dcmap:1 subprotocol="BFCP";max-time=60000
a=dcmap:2 subprotocol="MSRP";ordered=true;label="MSRP"
a=dcmap:3 label="Label 1";ordered=false;max-retr=5
a=dcmap:4 label="foo%09bar";ordered=true;max-time=15000;max-retr=3
```

Note: The last example (a=dcmap:4) shows a 'label' parameter value which contains one non-printable 'escaped-char' character (the tabulator character).

5.1.1.2. dcmap-stream-id Parameter

The 'dcmap-stream-id' parameter indicates the SCTP stream identifier within the SCTP association used to form the data channel.

5.1.1.3. label Parameter

The 'label' parameter indicates the name of the channel. It represents a label that can be used to distinguish, in the context of the WebRTC API, an RTCDataChannel object from other RTCDataChannel objects. This parameter maps to the 'Label' parameter defined in [I-D.ietf-rtcweb-data-protocol]. The 'label' parameter is optional. If it is not present, then its value defaults to the empty string.

Note: The empty string may also be explicitly used as 'label' value, such that 'label=""' is equivalent to the 'label' parameter not being present at all. [I-D.ietf-rtcweb-data-protocol] allows the DATA_CHANNEL_OPEN message's 'Label' value to be an empty string.

5.1.1.4. subprotocol Parameter

The 'subprotocol' parameter indicates which protocol the client expects to exchange via the channel. 'Subprotocol' is an optional parameter. If the 'subprotocol' parameter is not present, then its value defaults to the empty string.

5.1.1.5. max-retr Parameter

This parameter indicates that the data channel is partially reliable. The 'max-retr' parameter indicates the max times a user message will be retransmitted. The max-retr parameter is optional. If the max-retr parameter is not present, then the maximal number of retransmissions is determined as per the generic SCTP retransmission rules as specified in [RFC4960]. This parameter maps to the 'Number of RTX' parameter defined in [I-D.ietf-rtcweb-data-protocol].

5.1.1.6. max-time Parameter

This parameter indicates that the data channel is partially reliable. A user message will no longer be transmitted or retransmitted after a specified life-time given in milliseconds in the 'max-time' parameter. The max-time parameter is optional. If the max-time parameter is not present, then the generic SCTP retransmission timing rules apply as specified in [RFC4960]. This parameter maps to the

'Lifetime in ms' parameter defined in [I-D.ietf-rtcweb-data-protocol].

5.1.1.7. ordered Parameter

The 'ordered' parameter with value "true" indicates that DATA chunks in the channel MUST be dispatched to the upper layer by the receiver while preserving the order. The ordered parameter is optional and takes two values: "true" for ordered and "false" for unordered delivery with "true" as the default value. Any other value is ignored and default "ordered=true" is assumed. In the absence of this parameter "ordered=true" is assumed. This parameter maps to the ordered or unordered data channel types as defined in [I-D.ietf-rtcweb-data-protocol].

5.1.2. Sub-Protocol Specific Attributes

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

Each sub-protocol specific SDP attribute that would normally be used to negotiate the subprotocol using SDP is replaced with an attribute of the form "a=dcsa:stream-id original-attribute", where dcsa stands for "data channel sub-protocol attribute", stream-id is the SCTP stream identifier assigned to this sub-protocol instance, and original-attribute represents the contents of the sub-protocol related attribute to be included.

Formal Syntax:

Name: dcsa

Value: dcsa-value

Usage Level: media

Charset Dependent: no

Syntax:

```
dcsa-value      = stream-id SP attribute
attribute       = <from-RFC4566>
```

Example:

a=dcsa:2 accept-types:text/plain

Thus in the example above, the original attribute line "a=accept-types:text/plain" is represented by the attribute line "a=dcsa:2 accept-types:text/plain", which specifies that this instance of MSRP being transported on the sctp association using the data channel with stream id 2 accepts plain text files.

As opposed to the data channel "a=dcmap:" attribute parameters, these parameters are subject to offer/answer negotiation following the procedures defined in the sub-protocol specific documents.

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

Note: This document does not provide a complete specification of how to negotiate the use of a data channel to transport MSRP. Procedures specific to each sub-protocol such as MSRP will be documented elsewhere. The use of MSRP is only an example of how the generic procedures described herein might apply to a specific sub-protocol.

5.2. Procedures

5.2.1. Managing Stream Identifiers

If an SDP offer / answer exchange (could be the initial or a subsequent one) results in a UDP/DTLS/SCTP or TCP/DTLS/SCTP based media description being accepted, and if this SDP offer / answer exchange results in the establishment of a new SCTP association, then the SDP offerer owns the even SCTP stream ids of this new SCTP association and the answerer owns the odd SCTP stream identifiers.

If this "m" line is removed from the signaling session (its port number set to zero), and if usage of this or of a new UDP/DTLS/SCTP or TCP/DTLS/SCTP based "m" line is renegotiated later on, then the even and odd SCTP stream identifier ownership is redetermined as well as described above.

This specification allows simultaneous use of external and internal negotiation. However, a single stream is managed using one method at a time. Stream ids that are not currently used in SDP can be used for internal negotiation. Stream id allocation per SDP based external negotiation may not align with DTLS role based allocation. This could cause glare conditions when one side trying to do external negotiation on a stream id while the other end trying to open a data channel on the same stream id using internal negotiation. To avoid these glare conditions this specification recommends that the data channel stack user always selects stream ids per above described SDP offer / answer rule even when internal negotiation is used. To avoid glare conditions, it is possible to come up with a different stream id allocation scheme, but such schemes are outside the scope of this specification.

5.2.2. Negotiating Data Channel Parameters

Conveying a reliable data channel is achieved by including neither 'max-retr' nor 'max-time' in corresponding SDP offer's or answer's a=dcmap attribute line. Conveying a partially reliable data channel is achieved by including only one of 'max-retr' or 'max-time'. By definition max-retr and max-time are mutually exclusive, so only one of them can be present in a=dcmap. If an SDP offer contains both of these parameters then such an SDP offer will be rejected. If an SDP answer contains both of these parameters then the offerer may treat it as an error and may assume the associated SDP offer/answer failed and may take appropriate recovery actions. These recovery options are outside the scope of this specification.

The SDP answer shall echo the same subprotocol, max-retr, max-time, ordered parameters, if those were present in the offer, and may include a label parameter. They may appear in any order, which could be different from the SDP offer, in the SDP answer.

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

Data channel types defined in [I-D.ietf-rtcweb-data-protocol] are mapped to SDP in the following manner:

```
DATA_CHANNEL_RELIABLE
      a=dcmap:2 subprotocol="BFCP";label="channel 2"

DATA_CHANNEL_RELIABLE_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                ordered=0

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-retr=3

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-retr=3;ordered=0;

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-time=10000;

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
                max-time=10000; ordered=0
```

5.2.3. Opening a Data Channel

The procedure for opening a data channel using external negotiation starts with the agent preparing to send an SDP offer. If a peer receives an SDP offer before getting to send a new SDP offer with data channels that are to be externally negotiated, or loses an SDP offer glare resolution procedure in this case, it must wait until the ongoing SDP offer/answer completes before resuming the external negotiation procedure.

The agent that intends to send an SDP offer to create data channels through SDP-based external negotiation performs the following:

- Creates data channels using stream identifiers from the owned set (see Section 5.2.1).
- As described in Section 4.2.2, if the SCTP association is not yet established, then the newly created data channels are in the Connecting state, else if the SCTP association is already established, then the newly created data channels are in the Open state.
- Generates a new SDP offer. In the case of the browser based applications the browser generates the offer via the createOffer() API call [I-D.ietf-rtcweb-jsep].

- Determines the list of stream identifiers assigned to data channels opened through external negotiation.

- Completes the SDP offer with the dcmap and dcsa attributes needed, if any, for each externally-negotiated data channel, as described in Section 5.1 and in Section 5.2.2.

- Sends the SDP offer.

The peer receiving such an SDP offer performs the following:

- Parses and applies the SDP offer. Note that the typical parser normally ignores unknown SDP attributes, which includes data channel related attributes.

- Analyzes the channel parameters and sub-protocol attributes to determine whether to accept each offered data channel.

- For accepted data channels, it creates peer instances for the data channels with the agent using the channel parameters described in the SDP offer. Note that the agent is asked to create data channels with SCTP stream identifiers contained in the SDP offer if the SDP offer is accepted.

- As described in Section 4.2.2, if the SCTP association is not yet established, then the newly created data channels are in the Connecting state, else if the SCTP association is already established, then the newly created data channels are in the Open state.

- Generates an SDP answer.

- Completes the SDP answer with the dcmap and optional dcsa attributes needed for each externally-negotiated data channel, as described in Section 5.1 and in Section 5.2.2.

- Sends the SDP answer.

The agent receiving such an SDP answer performs the following:

- Closes any created data channels (whether in Connecting or Open state) for which the expected dcmap and dcsa attributes are not present in the SDP answer.

- Applies the SDP answer.

Any data channels in Connecting state are transitioned to the Open state when the SCTP association is established.

Each agent application MUST wait to send data until it has confirmation that the data channel at the peer is in the Open state. For WebRTC, this is when both data channel stacks have channel parameters instantiated. This occurs:

- At both peers when a data channel is created without an established SCTP association, as soon as the data channel stacks report that the data channel transitions to the Open state from the Connecting state.

- At the agent receiving an SDP offer for which there is an established SCTP association, as soon as it creates an externally negotiated data channel in the Open state based on information signaled in the SDP offer.

- At the agent sending an SDP offer to create a new externally negotiated data channel for which there is an established SCTP association, when it receives the SDP answer confirming acceptance of the data channel or when it begins to receive data on the data channel from the peer, whichever occurs first.

5.2.4. Closing a Data Channel

When the application requests the closing of a data channel that was externally negotiated, the data channel stack always performs an in-band SSN reset for this channel.

It is specific to the sub-protocol whether this closing must in addition be signaled to the peer via a new SDP offer/answer exchange.

The intention to close a data channel can be signaled by sending a new SDP offer which excludes the "a=dcmap:" and "a=dcsa:" attribute lines for the data channel. The port value for the "m" line SHOULD not be changed (e.g., to zero) when closing a data channel (unless all data channels are being closed and the SCTP association is no longer needed), since this would close the SCTP association and impact all of the data channels. If the answerer accepts the SDP offer then it MUST close those data channels whose "a=dcmap:" and "a=dcsa:" attribute lines were excluded from the received SDP offer, unless those data channels were already closed, and it MUST also exclude the corresponding attribute lines in the answer. In addition to that, the SDP answerer MAY exclude other data channels which were closed but not yet communicated to the peer. So, the offerer MUST inspect the answer to see if it has to close other data channels which are now not included in the answer.

If a new SDP offer/answer is used to close data channels then the data channel(s) SHOULD only be closed by the answerer/offerer after a successful SDP answer is sent/received.

This delayed closure is RECOMMENDED in order to handle cases where a successful SDP answer is not received, in which case the state of the session SHOULD be kept per the last successful SDP offer/answer.

If a client receives a data channel close indication (due to inband SSN reset or some other reason) without associated SDP offer then an SDP offer which excludes this closed data channel SHOULD be generated.

The application must also close any data channel that was externally negotiated, for which the stream identifiers are not listed in an incoming SDP offer.

A closed data channel using local close (SCTP reset), without an additional SDP offer/answer to close it, may be reused for a new data channel. This can only be done via new SDP offer/answer, describing the new sub-protocol and its attributes, only after the corresponding data channel close acknowledgement is received from the peer (i.e. SCTP reset of both incoming and outgoing streams is completed). This restriction is to avoid the race conditions between arrival of "SDP offer which reuses stream" with "SCTP reset which closes outgoing stream" at the peer

5.2.5. Various SDP Offer/Answer Scenarios and Considerations

SDP offer has no a=dcmap attributes

* Initial SDP offer: No data channel negotiated yet.

* Subsequent SDP offer: All the externally negotiated data channels must be closed now. The DTLS/SCTP association remains open for external or internal negotiation of data channels.

SDP answer has no a=dcmap attributes

* Initial SDP answer: Either the peer does not support dcmap attributes or it rejected all the data channels. In either case offerer closes all the externally negotiated data channels that were open at the time of initial offer. The DTLS/SCTP association will still be setup.

* Sub-sequent SDP answer: All the externally negotiated data channels must be closed now. The DTLS/SCTP association remains

open for future external or internal negotiation of data channels.

SDP offer has no a=dcsa attributes for a data channel.

* This is allowed and indicates there are no sub-protocol parameters to convey.

SDP answer has no a=dcsa attributes for a data channel.

* This is allowed and indicates there are no sub-protocol parameters to convey in the SDP answer. The number of dcsa attributes in the SDP answer does not have to match the number of dcsa attributes in the SDP offer.

6. Examples

SDP offer:

```
m=application 10001 UDP/DTLS/SCTP webrtc-datachannel
c=IN IP4 10.10.10.1
a=max-message-size:100000
a=sctp-port 5000
a=setup:actpass
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=dcmap:0 subprotocol="BFCP";label="BFCP"
```

SDP answer:

```
m=application 10002 UDP/DTLS/SCTP webrtc-datachannel
c=IN IP4 10.10.10.2
a=max-message-size:100000
a=sctp-port 5002
a=setup:passive
a=connection:new
a=fingerprint:SHA-1 \
    5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
```

Figure 1: Example 1

In the above example the SDP answerer rejected the data channel with stream id 0 either for explicit reasons or because it does not understand the a=dcmap attribute. As a result the offerer will close the data channel created with the external negotiation option. The SCTP association will still be setup over DTLS. At this point the offerer or the answerer may use internal negotiation to open data channels.

```
SDP offer:
  m=application 10001 UDP/DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.1
  a=max-message-size:100000
  a=sctp-port 5000
  a=setup:actpass
  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=dcmap:0 subprotocol="BFCP";label="BFCP"
  a=dcmap:2 subprotocol="MSRP";label="MSRP"
  a=dcsa:2 accept-types:message/cpim text/plain text/
  a=dcsa:2 path:msrp://alice.example.com:10001/2s93i93idj;dc

SDP answer:
  m=application 10002 UDP/DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.2
  a=max-message-size:100000
  a=sctp-port 5002
  a=setup:passive
  a=connection:new
  a=fingerprint:SHA-1 \
      5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
  a=dcmap:2 subprotocol="MSRP";label="MSRP"
  a=dcsa:2 accept-types:message/cpim text/plain
  a=dcsa:2 path:msrp://bob.example.com:10002/si438dsaodes;dc
```

Figure 2: Example 2

In the above example SDP offer contains data channels for BFCP and MSRP sub-protocols. SDP answer rejected BFCP and accepted MSRP. So, the offerer should close the data channel for BFCP and both offerer and answerer may start using MSRP data channel (after SCTP/DTLS association is setup). The data channel with stream id 0 is free and can be used for future internal or external negotiation.

Continuing on the earlier example in Figure 1.

```
Subsequent SDP offer:
  m=application 10001 UDP/DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.1
  a=max-message-size:100000
  a=sctp-port 5000
  a=setup:actpass
  a=connection:existing
  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=dcmap:4 subprotocol="MSRP";label="MSRP"
  a=dcsa:4 accept-types:message/cpim text/plain
  a=dcsa:4 path:msrp://alice.example.com:10001/2s93i93idj;dc

Subsequent SDP answer:
  m=application 10002 UDP/DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.2
  a=max-message-size:100000
  a=sctp-port 5002
  a=setup:passive
  a=connection:existing
  a=fingerprint:SHA-1 \
      5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
  a=dcmap:4 subprotocol="MSRP";label="MSRP"
  a=dcsa:4 accept-types:message/cpim text/plain
  a=dcsa:4 path:msrp://bob.example.com:10002/si438dsaodes;dc
```

Figure 3: Example 3

The above example is a continuation of the example in Figure 1. The SDP offer now removes the MSRP data channel with stream id 2, but opens a new MSRP data channel with stream id 4. The answerer accepted the entire offer. As a result the offerer closes the earlier negotiated MSRP related data channel and both offerer and answerer may start using new the MSRP related data channel.

7. Security Considerations

No security considerations are envisaged beyond those already documented in [RFC4566]

8. IANA Considerations

To be completed. As [I-D.ietf-rtcweb-data-protocol] this document should refer to IANA's WebSocket Subprotocol Name Registry defined in [RFC6455].

9. Acknowledgments

The authors wish to acknowledge the borrowing of ideas from other internet drafts by Salvatore Loreto, Gonzalo Camarillo, Peter Dunkley and Gavin Llewellyn, and to thank Christian Groves, Christer Holmberg, Paul Kyzivat, Jonathan Lennox, and Uwe Rauschenbach for their invaluable comments.

10. CHANGE LOG

10.1. Changes against ’draft-ietf-mmusic-data-channel-sdpneg-00’

- In Section 3 "WebRTC data channel" was defined as "A bidirectional channel consisting of paired SCTP outbound and inbound streams." Replacement of this definition with "Data channel: A WebRTC data channel as specified in [I-D.ietf-rtcweb-data-channel]", and consistent usage of "data channel" in the remainder of the document including the document’s headline."

- In Section 4 removal of following note: ’OPEN ISSUE: The syntax in [I-D.ietf-mmusic-sctp-sdp] may change as that document progresses. In particular we expect "webrtc-datachannel" to become a more general term.’

- Consistent usage of ’"m" line’ in whole document as per [RFC4566].

- In Section 5.1.1 removal of the example dcmap attribute line ’a=dcmap:2 subprotocol="BFCP";label="channel 2’ as there are already four examples right after the ABNF rules in Section 5.1.1.1. Corresponding removal of following related note: "Note: This document does not provide a complete specification of how to negotiate the use of a WebRTC data channel to transport BFCP. Procedures specific to each sub-protocol such as BFCP will be documented elsewhere. The use of BFCP is only an example of how the generic procedures described herein might apply to a specific sub-protocol."

- In Section 5.1.1 removal of following note: "Note: This attribute is derived from attribute "webrtc-DataChannel", which was defined in old version 03 of the following draft, but which was removed along with any support for SDP external negotiation in subsequent versions: [I-D.ietf-mmusic-sctp-sdp]."

- Insertion of following new sentence to the beginning of Section 5.1.1.1: "dcmap is a media level attribute having following ABNF syntax:"

- Insertion of new Section 5.1.1.2 containing the dcmap-stream-id specifying sentence, which previously was placed right before the formal ABNF rules. Removal of the sentence 'Stream is a mandatory parameter and is noted directly after the "a=dcmap:" attribute's colon' as this information is part of the ABNF specification.

- In Section 5.1.1.1 modification of the 'ordering-value' values from "0" or "1" to "true" or "false". Corresponding text modifications in Section 5.1.1.7.

- In Section 5.1.1.1 the ABNF definition of "quoted-string" referred to rule name "escaped-char", which was not defined. Instead a rule with name "escaped" was defined. Renamed that rule's name to "escaped-char".

- Insertion of a dedicated note right after the "a=dcmap:4" attribute example in Section 5.1.1.1 regarding the non-printable "escaped-char" character within the "label" value.

- In Section 5.1.2's second paragraph replacement of "sctp stream identifier" with "SCTP stream identifier".

- In first paragraph of Section 5.2.1 replacement of first two sentences 'For the SDP-based external negotiation described in this document, the initial offerer based "SCTP over DTLS" owns by convention the even stream identifiers whereas the initial answerer owns the odd stream identifiers. This ownership is invariant for the whole lifetime of the signaling session, e.g. it does not change if the initial answerer sends a new offer to the initial offerer.' with 'If an SDP offer / answer exchange (could be the initial or a subsequent one) results in a UDP/DTLS/SCTP or TCP/DTLS/SCTP based media description being accepted, and if this SDP offer / answer exchange results in the establishment of a new SCTP association, then the SDP offerer owns the even SCTP stream ids of this new SCTP association and the answerer owns the odd SCTP stream identifiers. If this "m" line is removed from the signaling session (its port number set to zero), and if usage of this or of a new UDP/DTLS/SCTP or TCP/DTLS/SCTP based "m" line is renegotiated later on, then the even and odd SCTP stream identifier ownership is redetermined as well as described above.'

- In Section 5.2.3 the first action of an SDP answerer, when receiving an SDP offer, was described as "Applies the SDP offer. Note that the browser ignores data channel specific attributes in the SDP." Replacement of these two sentences with "Parses and applies the SDP offer. Note that the typical parser normally ignores unknown SDP attributes, which includes data channel related attributes."

- In Section 5.2.3 the second sentence of the third SDP answerer action was "Note that the browser is asked to create data channels with stream identifiers not "owned" by the agent.". Replacement of this sentence with "Note that the agent is asked to create data channels with SCTP stream identifiers contained in the SDP offer if the SDP offer is accepted."

- In Section 5.2.4 the third paragraph began with "A data channel can be closed by sending a new SDP offer which excludes the dcmap and dcsa attribute lines for the data channel. The port value for the m line should not be changed (e.g., to zero) when closing a data channel (unless all data channels are being closed and the SCTP association is no longer needed), since this would close the SCTP association and impact all of the data channels. If the answerer accepts the SDP offer then it MUST also exclude the corresponding attribute lines in the answer. ..." Replacement of this part with "The intention to close a data channel can be signaled by sending a new SDP offer which excludes the "a=dcmap:" and "a=dcsa:" attribute lines for the data channel. The port value for the "m" line SHOULD not be changed (e.g., to zero) when closing a data channel (unless all data channels are being closed and the SCTP association is no longer needed), since this would close the SCTP association and impact all of the data channels. If the answerer accepts the SDP offer then it MUST close those data channels whose "a=dcmap:" and "a=dcsa:" attribute lines were excluded from the received SDP offer, unless those data channels were already closed, and it MUST also exclude the corresponding attribute lines in the answer."

- In Section 5.2.4 the hanging text after the third paragraph was "This delayed close is to handle cases where a successful SDP answer is not received, in which case the state of session should be kept per the last successful SDP offer/answer." Replacement of this sentence with "This delayed closure is RECOMMENDED in order to handle cases where a successful SDP answer is not received, in which case the state of the session SHOULD be kept per the last successful SDP offer/answer."

- Although dedicated to "a=dcmap" and "a=dcsa" SDP syntax aspects Section 5.1.1 contained already procedural descriptions related to data channel reliability negotiation. Creation of new Section 5.2.2 and moval of reliability negotiation related text to this new section.

10.2. Changes against 'draft-ejzak-mmusic-data-channel-sdpneg-02'

- Removal of note "[ACTION ITEM]" from section "subprotocol parameter". As [I-D.ietf-rtcweb-data-protocol] this document should refer to IANA's WebSocket Subprotocol Name Registry defined in [RFC6455].

- In whole document, replacement of "unreliable" with "partially reliable", which is used in [I-D.ietf-rtcweb-data-channel] and in [I-D.ietf-rtcweb-data-protocol] in most places.

- Clarification of the semantic if the "max-retr" parameter is not present in an a=dcmap attribute line. In section "max-retr parameter" the sentence "The max-retr parameter is optional with default value unbounded" was replaced with "The max-retr parameter is optional. If the max-retr parameter is not present, then the maximal number of retransmissions is determined as per the generic SCTP retransmission rules as specified in [RFC4960]".

- Clarification of the semantic if the "max-time" parameter is not present in an a=dcmap attribute line. In section "max-time parameter" the sentence "The max-time parameter is optional with default value unbounded" was replaced with "The max-time parameter is optional. If the max-time parameter is not present, then the generic SCTP retransmission timing rules apply as specified in [RFC4960]".

- In section "label parameter" the sentence "Label is a mandatory parameter." was removed and following new sentences (including the note) were added: "The 'label' parameter is optional. If it is not present, then its value defaults to the empty string. Note: The empty string may also be explicitly used as 'label' value, such that 'label=""' is equivalent to the 'label' parameter not being present at all. [I-D.ietf-rtcweb-data-protocol] allows the DATA_CHANNEL_OPEN message's 'Label' value to be an empty string."

- In section "subprotocol parameter" the sentence "Subprotocol is a mandatory parameter." was replaced with "'Subprotocol' is an optional parameter. If the 'subprotocol' parameter is not present, then its value defaults to the empty string."

- In the "Examples" section, in the first two SDP offer examples in the a=dcmap attribute lines 'label="BGCP"' was replaced with 'label="BFCP"'.

- In all examples, the "m" line proto value "DTLS/SCTP" was replaced with "UDP/DTLS/SCTP" and the "a=fmtp" attribute lines were

replaced with "a=max-message-size" attribute lines, as per draft-ietf-mmusic-sctp-sdp-12.

10.3. Changes against '-01'

- Formal syntax for dcmap and dcsa attribute lines.
- Making subprotocol as an optional parameter in dcmap.
- Specifying disallowed parameter combinations for max-time and max-retr.
- Clarifications on WebRTC data channel close procedures.

10.4. Changes against '-00'

- Revisions to identify difference between internal and external negotiation and their usage.
- Introduction of more generic terminology, e.g. "application" instead of "browser".
- Clarification of how "max-retr and max-time affect the usage of unreliable and reliable WebRTC data channels.
- Updates of examples to take into account the SDP syntax changes introduced with draft-ietf-mmusic-sctp-sdp-07.
- Removal of the SCTP port number from the a=dcmap and a=dcsa attributes as this is now contained in the a=sctp-port attribute, and as draft-ietf-mmusic-sctp-sdp-07 supports only one SCTP association on top of the DTLS connection.

11. References

11.1. Normative References

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

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

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[I-D.ietf-rtcweb-jsep]
Uberti, J., Jennings, C., and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-08 (work in progress), October 2014.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ietf-mmusic-sctp-sdp]
Holmberg, C., Loreto, S., and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-14 (work in progress), March 2015.

[WebRtcAPI]
Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD-webrtc-20130910, September 2013, <http://www.w3.org/TR/2013/WD-webrtc-20130910/>.

11.2. Informative References

[I-D.ietf-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", draft-ietf-rtcweb-data-protocol-09 (work in progress), January 2015.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for the Message Sessions Relay Protocol (MSRP)", RFC 4976, September 2007.

[RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.

[RFC6135] Holmberg, C. and S. Blau, "An Alternative Connection Model for the Message Session Relay Protocol (MSRP)", RFC 6135, February 2011.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.

Authors' Addresses

Keith Drage (editor)
Alcatel-Lucent
Quadrant, Stonehill Green, Westlea
Swindon
UK

Email: keith.drage@alcatel-lucent.com

Maridi R. Makaraju (Raju)
Alcatel-Lucent
2000 Lucent Lane
Naperville, Illinois
US

Email: Raju.Makaraju@alcatel-lucent.com

Juergen Stoetzer-Bradler
Alcatel-Lucent
Lorenzstrasse 10
D-70435 Stuttgart
Germany

Email: Juergen.Stoetzer-Bradler@alcatel-lucent.com

Richard Ejzak
Unaffiliated

Email: richard.ejzak@gmail.com

Jerome Marcon
Unaffiliated

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

M. Petit-Huguenin
Impedance Mismatch
A. Keranen
Ericsson
March 9, 2015

Using Interactive Connectivity Establishment (ICE) with Session Description Protocol (SDP) offer/answer and Session Initiation Protocol (SIP)
draft-ietf-mmusic-ice-sip-sdp-05

Abstract

This document describes how Interactive Connectivity Establishment (ICE) is used with Session Description Protocol (SDP) offer/answer and Session Initiation Protocol (SIP).

Table of Contents

1. Introduction

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 if ICE is not being used. 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 eight new SDP attributes -- the "candidate", "remote-candidates", "ice-lite", "ice-mismatch", "ice-ufrag", "ice-pwd", "ice-pacing", 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    = token
extension-att-value   = *VCHAR
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.

In some cases, e.g., for privacy reasons, an agent may not want to reveal the related address and port. In this case the address MUST be set to "0.0.0.0" (for IPv4 candidates) or "::" (for IPv6 candidates) and the port to zero.

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. For compatibility with the 512 character limitation for the STUN username attribute value and for bandwidth conservation considerations, the ice-ufrag attribute MUST NOT be longer than 32 characters when sending, but an implementation MUST accept up to 256 characters when receiving.

8.5. "ice-pacing" Attribute

The "ice-pacing" attribute indicates the desired connectivity check pacing, in milliseconds, for this agent (see Section 12.2 of [ICE-BIS]). The syntax is:

```
ice-pacing-att            = "ice-pacing" ":" pacing-value
pacing-value              = 1*10DIGIT
```

8.6. "ice-options" Attribute

The "ice-options" attribute is a session- and media-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
```

The existence of an ice-option can indicate that a certain extension is supported by the agent and will be used or that the extension is used only if the other agent is willing to use it too. In order to avoid ambiguity, documents defining new options must indicate which case applies to the defined extensions.

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:

- 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.

- 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:

- The agent MUST change the a=ice-ufrag and a=ice-pwd attributes in the answer.
- 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:

- 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)

- 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:

- 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.

- 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:

- 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).

- 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. Receiving the Answer for a Subsequent Offer

Some deployments of ICE include e.g. SDP-Modifying Signaling-only Back-to-Back User Agents (B2BUAs) [RFC7092] that modify the SDP body during the subsequent offer/answer exchange. With the B2BUA being ICE-unaware a subsequent answer might be manipulated and might not include ICE candidates although the initial answer did.

An example of a situation where such an "unexpected" answer might be experienced appears when such a B2BUA introduces a media server during call hold using 3rd party call-control procedures. Omitting further details how this is done this could result in an answer being received at the holding UA that was constructed by the B2BUA. With the B2BUA being ICE-unaware that answer would not include ICE candidates.

Receiving an answer without ICE attributes in this situation might be unexpected, but would not necessarily impair the user experience.

In addition to procedures for the expected answer, the following sections advice on how to recover from the unexpected situation.

9.3.1. Procedures for All Implementations

When receiving an answer within an existing session for a subsequent offer as specified in Section 9.1.2.2, an agent MUST verify ICE support as specified in Section 5.1.

9.3.1.1. ICE Restarts

If ICE support is indicated in the SDP answer, the agent MUST perform ICE restart procedures as specified in Section 9.4.

If ICE support is no longer indicated in the SDP answer, the agent MUST fall-back to RFC 3264 procedures and SHOULD NOT drop the dialog just because of missing ICE support. If the agent sends a new offer later on it SHOULD perform an ICE restart as specified in Section 9.1.1.1.

9.3.1.2. Existing Media Streams with ICE Running

If ICE support is indicated in the SDP answer, the agent MUST continue ICE procedures as specified in Section 9.4.1.4.

If ICE support is no longer indicated in the SDP answer, the agent MUST abort the ongoing ICE processing and fall-back to RFC 3264 procedures. The agent SHOULD NOT drop the dialog just because of missing ICE support. If the agent sends a new offer later on, it SHOULD perform an ICE restart as specified in Section 9.1.1.1.

9.3.1.3. Existing Media Streams with ICE Completed

If ICE support is indicated in the SDP answer and if the answer conforms to Section 9.2.2.3, the agent MUST remain in the ICE Completed state.

If ICE support is no longer indicated in the SDP answer, the agent MUST fall-back to RFC 3264 procedures and SHOULD NOT drop the dialog just because of this unexpected answer. Once the agent sends a new offer later on it MUST perform an ICE restart.

9.4. Updating the Check and Valid Lists

9.4.1. Procedures for Full Implementations

9.4.1.1. 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].

9.4.1.2. 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].

9.4.1.3. 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.

9.4.1.4. 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.

9.4.2. 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. 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:

$$Ta_i = (stun\_packet\_size / rtp\_packet\_size) * rtp\_ptime$$

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

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

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

RTO = MAX (100ms, Ta * (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 = MAX (100ms, Ta*N * (Num-Waiting + 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:

- The ALG does not modify the m and c lines or the rtcp attribute if they contain external addresses.
- 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-pacing Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-pacing

Long Form: ice-pacing

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) to indicate desired connectivity check pacing values.

Appropriate Values: See Section 8 of RFC XXXX.

16.1.8. ice-options Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-options

Long Form: ice-options

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 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 8.6; however, they are RECOMMENDED to be no longer than 20 characters. This is to reduce message sizes and allow for efficient parsing.

In RFC 5245 ICE options could only be defined at the session level. ICE options can now also be defined at the media level. This can be used when aggregating between different ICE agents in the same endpoint, but future options may require to be defined at the media-level. To ensure compatibility with legacy implementation, the media-level ICE options MUST be aggregated into a session-level ICE option. Because aggregation rules depend on the specifics of each option, all new ICE options MUST also define in their specification how the media-level ICE option values are aggregated to generate the value of the session-level ICE option.

The only ICE option defined at the time of publication is "rtp+ecn" [RFC6679]. The aggregation rule for this ICE options is that if all aggregated media using ICE contain a media-level "rtp+ecn" ICE option then an "rtp+ecn" ICE option MUST be inserted at the session-level. If one of the media does not contain the option, then it MUST NOT be inserted at the session-level.

A registration request MUST include the following information:

- The ICE option identifier to be registered
- Name, Email, and Address of a contact person for the registration
- Organization or individuals having the change control
- Short description of the ICE extension to which the option relates
- 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.

Thanks to Thomas Stach for the text in Section 9.3

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.

[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.

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, December 2013.

[ICE-BIS] Keranen, A. and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-rfc5245bis-04 (work in progress), March 2015.

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] Andreasen, 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.

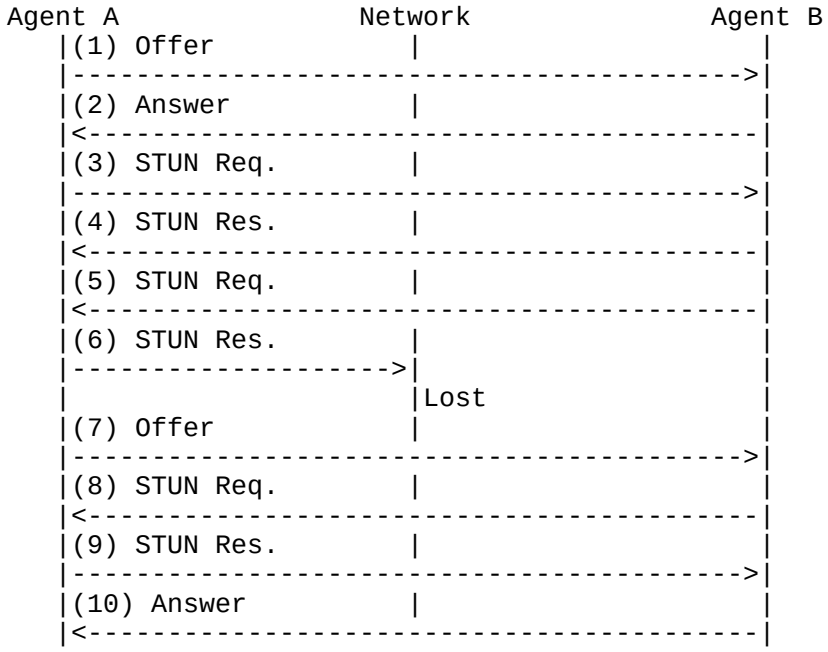


Figure 1: Race Condition Flow

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:

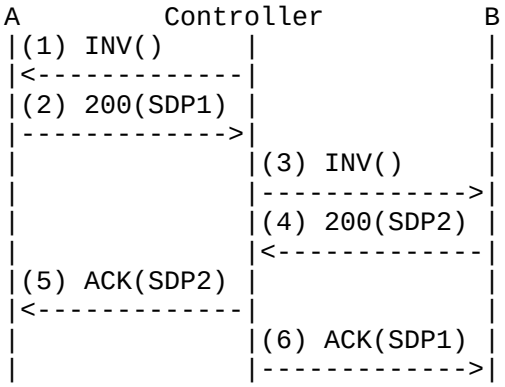


Figure 2: Role Conflict Flow

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: marc@petit-huguenin.org

Ari Keranen
Ericsson
Jorvas 02420
Finland

Email: ari.keranen@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 15, 2015

H. Alvestrand
Google

February 11, 2015

WebRTC MediaStream Identification in the Session Description Protocol
draft-ietf-mmusic-msid-08

Abstract

This document specifies a Session Description Protocol (SDP) Grouping mechanism for RTP media streams that can be used to specify relations between media streams.

This mechanism is used to signal the association between the SDP concept of "m-line" and the WebRTC concept of "MediaStream" / "MediaStreamTrack" using SDP signaling.

This document is a work item of the MMUSIC WG, whose discussion list is mmusic@ietf.org.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 15, 2015.

Table of Contents

1. Introduction

1.1. Structure Of This Document

This document adds a new Session Description Protocol (SDP) [RFC4566] mechanism that can associate application layer identifiers with the binding between media streams, attaching identifiers to the media streams and attaching identifiers to the groupings they form.

Section 1.2 gives the background on why a new mechanism is needed.

Section 2 gives the definition of the new mechanism.

Section 3 gives the definition of the msid-semantic field, which gives the possibility of using MSIDs with different semantics in the same SDP message.

Section 5 gives the application of the new mechanism for providing necessary semantic information for the association of MediaStreamTracks to MediaStreams in the WebRTC API [W3C.WD-webrtc-20120209].

1.2. Why A New Mechanism Is Needed

When media is carried by RTP [RFC3550], each RTP media stream is distinguished inside an RTP session by its SSRC; each RTP session is distinguished from all other RTP sessions by being on a different transport association (strictly speaking, 2 transport associations, one used for RTP and one used for RTCP, unless RTP/RTCP multiplexing [RFC5761] is used).

SDP gives a description based on m-lines. According to the model used in [I-D.ietf-rtcweb-jsep], each m-line describes exactly one media source, and if mulitple media sources are carried in an RTP

session, this is signalled using BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation]; if BUNDLE is not used, each media source is carried in its own RTP session.

There exist cases where an application using RTP and SDP needs to signal some relationship between RTP media streams that may be carried in either the same RTP session or different RTP sessions. For instance, there may be a need to signal a relationship between a video track and an audio track, and where the generator of the SDP does not yet know if they will be carried in the same RTP session or different RTP sessions.

The SDP grouping framework [RFC5888] can be used to group m-lines. However, there is sometimes the need for an application to specify some application-level information about the association between the m-line and the group. This is not possible using the SDP grouping framework.

1.3. Application to the WEBRTC MediaStream

The W3C WebRTC API specification [W3C.WD-webrtc-20120209] specifies that communication between WebRTC entities is done via MediaStreams, which contain MediaStreamTracks. A MediaStreamTrack is generally carried using a single SSRC in an RTP session (forming an RTP media stream. The collision of terminology is unfortunate.) There might possibly be additional SSRCs, possibly within additional RTP sessions, in order to support functionality like forward error correction or simulcast. This complication is ignored below.

In the RTP specification, media streams are identified using the SSRC field. Streams are grouped into RTP Sessions, and also carry a CNAME. Neither CNAME nor RTP session correspond to a MediaStream. Therefore, the association of an RTP media stream to MediaStreams need to be explicitly signaled.

WebRTC defines a mapping (documented in [I-D.ietf-rtcweb-jsep]) where one SDP m-line is used to describe each MediaStreamTrack, and that the BUNDLE mechanism [I-D.ietf-mmusic-sdp-bundle-negotiation] is used to group MediaStreamTracks into RTP sessions. Therefore, the need is to specify the ID of a MediaStreamTrack and its associated MediaStream for each m-line, which can be accomplished with a media-level SDP attribute.

This usage is described in Section 5.

2. The Msid Mechanism

This document defines a new SDP [RFC4566] media-level "msid" attribute. This new attribute allows endpoints to associate RTP media streams that are carried in the same or different m-lines. The attribute also allows application-specific information to the association.

The value of the "msid" attribute consists of an identifier and optional application-specific data.

The name of the attribute is "msid".

The value of the attribute is specified by the following ABNF [RFC5234] grammar:

```
msid-value = msid-id [ SP msid-appdata ]
msid-id = 1*64token-char ; see RFC 4566
msid-appdata = 1*64token-char  ; see RFC 4566
```

An example msid value for a group with the identifier "examplefoo" and application data "examplebar" might look like this:

```
msid:examplefoo examplebar
```

The identifier is a string of ASCII characters that are legal in a "token", consisting of between 1 and 64 characters. It MUST be unique among the identifier values used in the same SDP session. It is RECOMMENDED that it is generated using a random-number generator.

Application data is carried on the same line as the identifier, separated from the identifier by a space.

The identifier uniquely identifies a group within the scope of an SDP description.

There may be multiple msid attributes in a single media description. There may also be multiple media descriptions that have the same value for identifier and application data.

Endpoints can update the associations between RTP media streams as expressed by msid attributes at any time; the semantics and restrictions of such grouping and ungrouping are application dependent.

3. The Msid-Semantic Attribute

A session-level attribute is defined for signaling the semantics associated with an msid grouping. This allows msid groupings with different semantics to coexist.

This OPTIONAL attribute gives the group identifier and its group semantic; it carries the same meaning as the ssrc-group-attr of RFC 5576 section 4.2, but uses the identifier of the group rather than a list of SSRC values.

This attribute MUST be present if "a=msid" is used.

An empty list of identifiers is an indication that the sender supports the indicated semantic, but has no msid groupings of the given type in the present SDP.

An identifier of "*" is an indication that all "a=msid" lines in the SDP have this specific semantic. If "*" is not used, each msid-id in the SDP MUST appear in one and only one "msid-semantic" line.

The name of the attribute is "msid-semantic".

The value of the attribute is given by the following ABNF:

```
msid-semantic-value = msid-semantic msid-list
msid-semantic = token ; see RFC 4566
msid-list = *(" " msid-id) / " *"
```

The semantic field holds values from the IANA registriy "Semantics for the msid-semantic SDP attribute" (which is defined in Section 6).

An example msid-semantic might look like this, if a semantic LS was registered by IANA for the same purpose as the existing LS grouping semantic:

```
a=msid-semantic:LS xyzzy forolow
```

This means that the SDP description has two lip sync groups, with the group identifiers xyzzy and forolow, respectively.

The msid-semantic attribute can occur more than once, but MUST NOT occur more than once with the same msid-semantic value.

4. Generic SDP Offer/Answer Procedures

In accordance with guidance on definitions of SDP extensions, this section gives the generic procedures that have to be followed by all implementations of Msid, independent of which semantics they support.

Note that the use of msid is not negotiated; each side declares what semantics it uses. This means that an offerer has to be willing and able to take appropriate action if the other side does not wish to use the semantic, and an answerer adding new semantics to an answer has to be willing and able to deal with the offerer not wishing to use that semantic.

4.1. Generating the Initial Offer

An entity wishing to use an MSID semantic MUST add one or more "msid-semantic" attributes to its session level attributes, indicating the MSID semantic it wishes to have available..

4.2. Answerer Processing of the Offer

If an "msid-semantic" attribute is present in the offer, and the answerer wishes to use the indicated semantic, the offerer MUST follow the procedures described for that semantic.

4.3. Generating the Answer

An entity wishing to use an MSID semantic MUST add one or more "msid-semantic" attributes to its session level attributes, indicating the MSID semantic it wishes to have available. If the answerer does not wish to use one or more of the semantics indicated in the offer, the answerer MUST NOT include "msid-semantic" lines indicating these semantics in the answer.

4.4. Offerer Processing of the Answer

If an "msid-semantic" attribute is present in the answer, and the offerer wishes to use the indicated semantic, the offerer MUST follow the procedures described for that semantic. The offerer MUST follow the procedures for all semantics that were indicated in its offer and were also present in the answer.

5. Applying Msid to WebRTC MediaStreams

This section creates a new semantic for use with the framework defined in Section 2, to be used for associating m-lines representing MediaStreamTracks within MediaStreams as defined in [W3C.WD-webrtc-20120209].

In the Javascript API, each MediaStream and MediaStreamTrack has an "id" attribute, which is a DOMString.

The semantic token for this semantic is "WMS" (short for WebRTC Media Stream).

The value of the "identifier" field in the msid consists of the "id" attribute of a MediaStream, as defined in its WebIDL specification.

The value of the "appdata" field in the msid consists of the "id" attribute of a MediaStreamTrack, as defined in its WebIDL specification.

If two different m-lines have MSID attributes with the same value for identifier and appdata, it means that these two m-lines are both intended for the same MediaStreamTrack. So far, no semantic for such a mixture have been defined, but this specification does not forbid the practice.

When an SDP description is updated, a specific msid "identifier" continues to refer to the same MediaStream, and a specific "appdata" to the same MediaStreamTrack. Once negotiation has completed on a session, there is no memory apart from the currently valid SDP descriptions; if an msid "identifier" value disappears from the SDP and appears in a later negotiation, it will be taken to refer to a new MediaStream.

The following are the rules for handling updates of the list of m-lines and their msid values.

- When a new msid "identifier" value occurs in the description, the recipient can signal to its application that a new MediaStream has been added.

- When a description is updated to have more media sections with the same msid "identifier" value, but different "appdata" values, the recipient can signal to its application that new MediaStreamTracks have been added to the MediaStream.

- When a description is updated to no longer list the msid attribute on a specific media description, the recipient can signal to its application that the corresponding MediaStreamTrack has ended.

In addition to signaling that the track is closed when its msid attribute disappears from the SDP, the track will also be signaled as being closed when all associated SSRCs have disappeared by the rules of [RFC3550] section 6.3.4 (BYE packet received) and 6.3.5 (timeout), and when the corresponding media section is disabled by setting the

port number to zero. Changing the direction of the media section (by setting "sendonly", "recvonly" or "inactive" attributes) will not close the MediaStreamTrack.

The association between SSRCs and m-lines is specified in [I-D.ietf-rtcweb-jsep].

5.1. Handling of non-signalled tracks

Entities that do not use the WMS semantic will not send "msid-semantic:WMS". This means that there will be some incoming RTP packets that the recipient has no predefined MediaStream id value for.

Note that this handling is triggered by incoming RTP packets, not by SDP negotiation.

Handling will depend on whether or not the msid-semantic:WMS attribute is present. There are two cases:

- No "msid-semantic:WMS" attribute is present. The SDP session is assumed to be a backwards-compatible session. All incoming media, on all m-lines that are part of the SDP session, are assumed to belong to tracks of the same media stream (the "default media stream"). The identifier of this media stream and of the media stream track is a randomly generated string; the WebIDL "label" attribute of this media stream will be set to "Non-WMS stream".

- An "msid-semantic:WMS" attribute is present. In this case, the sender implements the WMS semantic, and the packets are either caused by a bug or by timing skew between the arrival of the media packets and the SDP description. These packets MAY be discarded, or they MAY be buffered for a while in order to allow immediate startup of the media stream when the SDP description is updated. The arrival of media packets MUST NOT cause a new MediaStreamTrack to be signaled.

If an entity wishing to use the WMS semantic sends a description, it MUST include the msid-semantic:WMS attribute, even if no media streams are sent. This allows us to distinguish between the case of no media streams at the moment and the case of SDP generated by an entity that wishes to use the backwards-compatible mechanism.

It follows from the above that the media receiver implmementing the WMS semantic must have the SDP of the other party before it can decide correctly which of the two cases described above applies. RTP media packets that arrive before the remote party’s SDP MUST be

buffered or discarded, and MUST NOT cause a new MediaStreamTrack to be signalled.

It follows from the above that media stream tracks in the "default" media stream cannot be closed by removing the msid attribute; the application must instead signal these as closed when the SSRC disappears according to the rules of RFC 3550 section 6.3.4 and 6.3.5 or by disabling the m-line by setting its port to zero.

5.2. Detailed Offer/Answer Procedures

These procedures are given in terms of RFC 3264-recommended sections. They describe the actions to be taken in terms of MediaStreams and MediaStreamTracks; they do not include event signalling inside the application, which is described in JSEP.

They are specifically applicable to the WMS semantic; other semantics will have their own consideration.

5.2.1. Generating the initial offer

For each media section in the offer, if there is an associated MediaStreamTrack, the offerer adds one "a=msid" attribute to the section for each MediaStream with which the MediaStreamTrack is associated. The "identifier" field of the attribute is set to the WebIDL "id" attribute of the MediaStream, and the "appdata" field is set to the WebIDL "id" attribute of the MediaStreamTrack.

The offerer adds an "msid-semantic:WMS" field to the session-level headers, and appends to it either a list of all the identifiers used in the offer, or the single character "*".

5.2.2. Parsing the initial offer

For each media section in the offer, and for each "a=msid" attribute in the media section where the "msid-id" is associated with the "WMS" semantic, the receiver of the offer will perform the following steps:

- Extract the "appdata" field of the "a=msid" attribute
- Check if a MediaStreamTrack with the same WebIDL "id" attribute as the "appdata" field already exists, and is not in the "ended" state. If it is not found, create it.
- Extract the "identifier" field of the "a=msid" attribte.
- Check if a MediaStream with the same WebIDL "id" attribute already exists. If not, create it.

- Add the MediaStreamTrack to the MediaStream

5.2.3. Generating the answer

The answer is generated in exactly the same manner as the offer.

This includes adding a "msid-semantic:WMS" attribute in the session-level headers, independent of whether or not such a header was present in the offer.

5.2.4. Offerer processing of the answer

The answer is processed in exactly the same manner as the offer.

5.2.5. Modifying the session

On subsequent exchanges, precisely the same procedure as for the initial offer/answer is followed, but with one additional step in the parsing of the offer and answer:

- For each MediaStreamTrack that has been created as a result of previous offer/answer exchanges, and is not in the "ended" state, check to see if there is still an "a=msid" attribute in the present SDP whose "appdata" field is the same as the WebIDL "id" attribute of the track.

- If no such attribute is found, close the MediaStreamTrack. This will set its state to "ended".

6. IANA Considerations

6.1. Attribute registration in existing registries

This document requests IANA to register the "msid" attribute in the "att-field (media level only)" registry within the SDP parameters registry, according to the procedures of [RFC4566]

The required information for "msid" is:

- Contact name, email: IETF, contacted via mmusic@ietf.org, or a successor address designated by IESG

- Attribute name: msid

- Long-form attribute name: Media stream group Identifier

- Subject to charset: The attribute value contains only ASCII characters, and is therefore not subject to the charset attribute.

- Purpose: The attribute gives an association over a set of m-lines. For example, it can be used to signal the relationship between a WebRTC MediaStream and a set of m-lines.

- Appropriate values: The details of appropriate values are given in RFC XXXX.

This document requests IANA to register the "msid-semantic" attribute in the "att-field (session level) registry within the SDP parameters registry, according to the same procedures.

The required information is:

- Contact name, email: IETF, contacted via mmusic@ietf.org, or a successor address designated by IESG

- Attribute name: msid-semantic

- Long-form attribute name: Msid group semantic identifier

- Subject to charset: The attribute value contains only ASCII characters, and is therefore not subject to the charset attribute.

- Purpose: The attribute gives the semantics of an association over a set of m-lines.

- Appropriate values: The details are given in RFC XXXX.

6.2. New registry creation

This document requests IANA to create a new registry called "Semantics for the msid-semantic SDP attribute" in the "Session Description Protocol (SDP) Parameters" group. This registry operates on the Expert Review policy [RFC5226]. Usage of the registry is expected to be low, so the expert should feel free to consult widely if a new request ever comes in.

This document requests IANA to register the "WMS" semantic within this new registry.

The required information is:

- Description: WebRTC Media Stream, as given in RFC XXXX.

- Token: WMS

- Standards track reference: RFC XXXX

IANA is requested to replace "RFC XXXX" with the RFC number of this document upon publication.

7. Security Considerations

An adversary with the ability to modify SDP descriptions has the ability to switch around tracks between media streams. This is a special case of the general security consideration that modification of SDP descriptions needs to be confined to entities trusted by the application.

If implementing buffering as mentioned in Section 5.1, the amount of buffering should be limited to avoid memory exhaustion attacks.

No other attacks have been identified that depend on this mechanism.

8. Acknowledgements

This note is based on sketches from, among others, Justin Uberti and Cullen Jennings.

Special thanks to Flemming Andreassen, Miguel Garcia, Martin Thomson, Ted Hardie, Adam Roach and Paul Kyzivat for their work in reviewing this draft, with many specific language suggestions.

9. References

9.1. Normative References

[I-D.ietf-rtcweb-jsep] Uberti, J., Jennings, C., and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-08 (work in progress), October 2014.

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[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.

[W3C.WD-webrtc-20120209] Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120209, February 2012, <http://www.w3.org/TR/2012/WD-webrtc-20120209>.

9.2. Informative References

[I-D.ietf-mmusic-sdp-bundle-negotiation] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-07 (work in progress), April 2014.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

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

Appendix A. Design considerations, rejected alternatives

This appendix should be deleted before publication as an RFC.

One suggested mechanism has been to use CNAME instead of a new attribute. This was abandoned because CNAME identifies a synchronization context; one can imagine both wanting to have tracks from the same synchronization context in multiple MediaStreams and wanting to have tracks from multiple synchronization contexts within one MediaStream (but the latter is impossible, since a MediaStream is defined to impose synchronization on its members).

Another suggestion has been to put the msid value within an attribute of RTCP SR (sender report) packets. This doesn’t offer the ability to know that you have seen all the tracks currently configured for a media stream.

Appendix B. Change log

This appendix should be deleted before publication as an RFC.

B.1. Changes from alvestrand-rtcweb-msid-00 to -01

Added track identifier.

Added inclusion-by-reference of draft-lennox-mmusic-source-selection for track muting.

Some rewording.

B.2. Changes from alvestrand-rtcweb-msid-01 to -02

Split document into sections describing a generic grouping mechanism and sections describing the application of this grouping mechanism to the WebRTC MediaStream concept.

Removed the mechanism for muting tracks, since this is not central to the MSID mechanism.

B.3. Changes from alvestrand-rtcweb-msid-02 to mmusic-msid-00

Changed the draft name according to the wishes of the MMUSIC group chairs.

Added text indicting cases where it’s appropriate to have the same appdata for multiple SSRCs.

Minor textual updates.

B.4. Changes from alvestrand-mmusic-msid-00 to -01

Increased the amount of explanatory text, much based on a review by Miguel Garcia.

Removed references to BUNDLE, since that spec is under active discussion.

Removed distinguished values of the MSID identifier.

B.5. Changes from alvestrand-mmusic-msid-01 to -02

Changed the order of the "msid-semantic: " attribute’s value fields and allowed multiple identifiers. This makes the attribute useful as a marker for "I understand this semantic".

Changed the syntax for "identifier" and "appdata" to be "token".

Changed the registry for the "msid-semantic" attribute values to be a new registry, based on advice given in Atlanta.

B.6. Changes from alvestrand-mmusic-msid-02 to ietf-mmusic-00

Updated terminology to refer to m-lines rather than RTP sessions when discussing SDP formats and the ability of other linking mechanisms to refer to SSRCs.

Changed the "default" mechanism to return independent streams after considering the synchronization problem.

Removed the space from between "msid-semantic" and its value, to be consistent with RFC 5576.

B.7. Changes from mmusic-msid-00 to -01

Reworked msid mechanism to be a per-m-line attribute, to align with draft-roach-mmusic-unified-plan.

B.8. Changes from mmusic-msid-01 to -02

Corrected several missed cases where the word "ssrc" was not changed to "M-line".

Added pointer to unified-plan (which should be moved to point to -jsep)

Removed suggestion that ssrc-group attributes can be used with "msid-semantic", it is now only the msid-semantic registry.

B.9. Changes from mmusic-msid-02 to -03

Corrected even more cases where the word "ssrc" was not changed to "M-line".

Added the functionality of using an asterisk (*) in the msid-semantic line, in order to remove the need for listing all msids in the msid-semantic line whne only one msid-semantic is in use.

Removed some now-unnecessary text.

B.10. Changes from mmusic-msid-03 to -04

Changed title to reflect focus on WebRTC MediaStreams

Added a section on receiver-side media stream control, using the "msid-control" attribute.

B.11. Changes from -04 to -05

Removed the msid-control section after WG discussion.

Removed some text that seemed only to pertain to resolved issues.

B.12. Changes from -05 to -06

Addressed issues found in Fleming Andreassen’s review

Referenced JSEP rather than unified-plan for the M-line mapping model

Relaxed MSID definition to allow "token-char" in values rather than a-z 0-9 hyphen; tightened ABNF by adding length description to it.

Deleted discussion of abandoned alternatives, as part of preparing for publication.

Added a "detailed procedures" section to the WMS semantics description.

Added IANA registration of the "msid-semantic" attribute.

B.13. Changes from -06 to -07

Changed terminology from referring to "WebRTC device" to referring to "entities that implement the WMS semantic".

Changed names for ABNF constructions based on a proposal by Paul Kyzivat.

Included a section on generic offer/answer semantics.

B.14. Changes from -07 to -08

Removed Appendix B that described the (now obsolete) ssrc-specific usage of MSID.

Adopted a restructuring of the IANA section based on a suggestion from Martin Thomson.

A number of text and ABNF clarifications based on suggestions from Ted Hardie, Paul Kyzivat and Adam Roach.

Changed the "non-signalled track handling" to create a single stream with multiple tracks again, according to discussions at TPAC in November 2014

Author's Address

Harald Alvestrand
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2015

K. Drage, Ed.
M. Makaraju
J. Stoetzer-Bradler
Alcatel-Lucent
R. Ejzak
J. Marcon
Unaffiliated

March 8, 2015

MSRP over Data Channels
draft-ietf-mmusic-msrp-usage-data-channel-01

Abstract

This document specifies how the Message Session Relay Protocol (MSRP) can be instantiated as a data channel sub-protocol, using the the SDP offer/answer exchange-based external negotiation defined in [I-D.ietf-mmusic-data-channel-sdpneg]. Two network configurations are documented: a WebRTC end-to-end configuration (connecting two MSRP over data channel endpoints), and a gateway configuration (connecting an MSRP over data channel endpoint with an MSRP over TCP endpoint).

Status of This Memo

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

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

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

This Internet-Draft will expire on September 9, 2015.

Table of Contents

1. Introduction

The Message Session Relay Protocol (MSRP) [RFC4975] is a protocol for transmitting a series of related instant messages in the context of a session. In addition to instant messaging, MSRP can also be used for image sharing or file transfer. MSRP is currently defined to work over TCP and TLS connections.

This document defines the negotiation and transport of this MSRP protocol over data channels, where a data channel is a bi-directional communication channel running on top of SCTP/DTLS (as per [I-D.ietf-rtcweb-data-channel]) and where MSRP is instantiated as a sub-protocol of this data channel.

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

- provides to applications a proven protocol enabling instant messaging, file transfer, image sharing
- integrates those features with other RTCWeb voice, video and data features
- leverages the SDP-based negotiation already defined for MSRP
- allows the interworking with MSRP endpoints running on a TCP or TLS connection

Considering an MSRP endpoint being an MSRP application that uses data channel from WebRTC specifications [I-D.ietf-rtcweb-data-channel], this document describes two configurations where the other endpoint is respectively either another MSRP over data channel endpoint (e.g., a WebRTC application) or an MSRP endpoint using either TCP or TLS transport.

2. Conventions

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

3. Terminology

This document uses the following terms:

Data channel: A WebRTC data channel as specified in [I-D.ietf-rtcweb-data-channel].

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

External negotiation: Data channel negotiation based on out-of-band or in-band mechanisms other than the Data Channel Establishment Protocol specified in [I-D.ietf-rtcweb-data-protocol].

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

Out-of-band: Transmission through the call control signaling path, e.g., using JSEP [I-D.ietf-rtcweb-jsep] and the SDP Offer/Answer model [RFC3264].

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

4. Principles

4.1. MSRP Data Channel

In this document, an MSRP data channel is a data channel for which the instantiated sub-protocol is "msrp", and where the MSRP-related negotiation is done as part of the SDP-based external negotiation method defined in [I-D.ietf-mmusic-data-channel-sdpneg].

4.2. Session Mapping

In this design, the MSRP session maps to the SCTP association and the "SCTP stream pair" assigned to the data channel, and each MSRP session maps to one data channel exactly.

4.3. MSRP URI

This document extends the MSRP URI syntax [RFC4975] by defining the new transport parameter value "dc":

```
transport  /= "dc" / 1*ALPHANUM
              ; Add "dc" to existing transports per [RFC4975]
```

4.4. msrp-scheme

The msrp-scheme portion of the MSRP-URI that represents an MSRP data channel endpoint (used in the SDP path attribute and in the MSRP message headers) is always "msrps", which indicates that the MSRP data channel is always secured using DTLS.

5. End-to-End Configuration

This section describes the network configuration where each MSRP endpoint is running MSRP over a data channel.

5.1. Basic MSRP Support

5.1.1. Session Negotiation

5.1.1.1. Use of dcmap Attribute

The SDP offer shall include a dcmap attribute line (defined in [I-D.ietf-mmusic-data-channel-sdpneg]), within the media description for the SCTP association for each MSRP data channel session to be negotiated.

The attribute includes the following data channel parameters:

- "label=" labelstring
- "subprotocol=" "MSRP"

The labelstring is set by the MSRP application according to [I-D.ietf-mmusic-data-channel-sdpneg]. The max-retr, max-time and ordered parameters shall not be used.

Rest of the SDP offer/answer procedures are per [I-D.ietf-mmusic-data-channel-sdpneg]

The following is an example of the dcmap attribute for an MSRP session to be negotiated (on default SCTP port 5000) with stream=2 and label="chat":

```
a=dcmap:2 label="chat";subprotocol="MSRP"
```

5.1.1.2. Use of dcsa Attribute

The SDP offer shall also include a dcsa attribute line (defined in [I-D.ietf-mmusic-data-channel-sdpneg]) within the media description for the SCTP association for each MSRP-specific SDP attribute to be negotiated for each MSRP data channel being negotiated.

The MSRP-specific items that can be negotiated include at least all of the following well-known attributes:

- defined in [RFC4975]: "path", "accept-types", "accept-wrapped-types", "max-size"

- defined in [RFC4566]: "sendonly", "recvonly", "inactive", and "sendrecv"
- defined in [RFC6135]: "setup"
- defined in [RFC6714]: "msrp-cema"
- defined in [RFC5547]: all the parameters related to MSRP file transfer. See Section 5.2.

The msrp-cema attribute shall be assumed to be present for every MSRP session using data channel transport, so the inclusion of the msrp-cema attribute is optional. This ensures that the data channel transport for the MSRP session is established without using the path attribute.

The SDP answer shall include zero or more corresponding dcsa attribute lines for each negotiated MSRP session, according to the MSRP-specific attribute negotiation rules in the corresponding specifications.

A new SDP offer/answer may update the MSRP subprotocol attributes while keeping the same subprotocol a=dcmap description. The semantics for newly negotiated MSRP subprotocol attributes are per [RFC4975]

5.1.1.3. Example SDP Negotiation

The following is an example of an "m" line for data channels in an SDP offer that includes the attributes needed to establish two MSRP sessions: one for chat and one for file transfer. The example is derived from a combination of examples in [RFC4975] and [RFC5547].

```
m=application 54111 UDP/DTLS/SCTP webrtc-datachannel
c=IN IP4 79.97.215.79
a=max-message-size:100000
a=sctp-port 5000
a=dcmap:1 label="chat";subprotocol="MSRP"
a=dcsa:1 accept-types:message/cpim text/plain
a=dcsa:1 path:msrps://bob.example.com:54111/si438dsaodes;dc
a=dcmap:2 label="file transfer";subprotocol="MSRP"
a=dcsa::2 sendonly
a=dcsa:2 accept-types:message/cpim
a=dcsa:2 accept-wrapped-types:*
a=dcsa:2 path:msrps://bob.example.com:54111/jshA7we;dc
a=dcsa:2 file-selector:name:"My cool picture.jpg" \
     type:image/jpeg size:32349 hash:sha-1: \
     72:24:5F:E8:65:3D:DA:F3:71:36:2F:86:D4:71:91:3E:E4:A2:CE:2E
a=dcsa:2 file-transfer-id:vBnG916bdberum2fFEABR1FR3ExZMUrd
a=dcsa:2 file-disposition:attachment
a=dcsa:2 file-date:creation:"Mon, 15 May 2006 15:01:31 +0300"
a=dcsa:2 file-icon:cid:id2@bob.example.com
a=dcsa:2 file-range:1-32349
```

5.1.2. Session Opening

The active MSRP endpoint does not use the path attribute to open a transport connection to its peer. Instead, it uses the data channel established for this MSRP session by the generic data channel opening procedure defined in [I-D.ietf-mmusic-data-channel-sdpneg].

As soon as this data channel is opened, the MSRP session is actually opened by the active MSRP endpoint which sends an MSRP SEND message (empty or not) to the other MSRP endpoint. The msrp-cema attribute is implicitly associated with every MSRP session using data channel transport.

5.1.3. Data Framing

Each text-based MSRP message is sent on the corresponding SCTP stream using standard MSRP framing and chunking procedures, as defined in [RFC4975], with each MSRP chunk delivered in a single SCTP user message.

5.1.4. Data Sending and Reporting

Data sending and reporting procedures shall conform to RFC 4975.

5.1.5. Session Closing

Closing of an MSRP session is done using the generic data channel closing procedure defined in [I-D.ietf-mmusic-data-channel-sdpneg].

The port value for the "m" line should not be changed (e.g., to zero) when closing an MSRP session (unless all data channels are being closed and the SCTP association is no longer needed), since this would close the SCTP association and impact all of the data channels. In all cases in [RFC4975] where the procedure calls for setting the port to zero for the MSRP "m" line in an SDP offer for TCP transport, the SDP offerer of an MSRP session with data channel transport shall remove the corresponding dcmap and dcsa attributes.

The SDP answerer must ensure that no dcmap or dcsa attributes are present in the SDP answer if no corresponding attributes are present in the received SDP offer.

5.2. Support for MSRP File Transfer Function

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

- As an MSRP session maps to one data channel, a file transfer session maps also to one data channel.
- SDP attributes specified in [RFC5547] for a file transfer "m" line are embedded as subprotocol-specific attributes using the syntax defined in [I-D.ietf-mmusic-data-channel-sdpneg].
- Once the file transfer is complete, the same data channel MAY be reused for another file transfer.

6. Gateway Configuration

This section describes the network configuration where one MSRP endpoint uses data channels as MSRP transport, the other MSRP endpoint uses TLS/TCP connections as MSRP transport, and the two MSRP endpoints interwork via an MSRP gateway.

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

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

When the gateway performs transport level interworking between MSRP endpoints, all of the procedures in Section 5 apply to each peer, with the following additions:

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

7. Security Considerations

To be completed.

8. IANA Considerations

To be completed.

9. Acknowledgments

The authors wish to acknowledge the borrowing of ideas from another internet draft by Peter Dunkley and Gavin Llewellyn, and to thank Christian Groves, Christer Holmberg, Paul Kyzivat, Jonathan Lennox, Uwe Rauschenbach and Keith Drage for their invaluable comments.

10. CHANGE LOG

10.1. Changes against ’draft-ietf-mmusic-msrp-usage-data-channel-00’

- Additional reference to [I-D.ietf-mmusic-data-channel-sdpneg] in list of normative references.

- Replacement of previous document title "MSRP over SCTP/DTLS data channels" with "MSRP over Data Channels" in order to align with the terminology used in [I-D.ietf-mmusic-data-channel-sdpneg].

- In Section 3 "WebRTC data channel" was defined as "A bidirectional channel consisting of paired SCTP outbound and inbound streams." Replacement of this definition with "Data channel: A WebRTC data channel as specified in [I-D.ietf-rtcweb-data-channel]", and consistent usage of either "data channel" or "MSRP data channel" in the remainder of the document."

- In the introduction replacement of references to [I-D.ietf-rtcweb-data-protocol] with a reference to [I-D.ietf-rtcweb-data-channel].

- Consistent usage of ’"m" line’ in whole document as per [RFC4566].

- In the gateway configuration section (Section 6) replacement of the first sentence "This section describes the network configuration where one endpoint runs MSRP over a WebRTC SCTP/DTLS connection, the other MSRP endpoint runs MSRP over one or more TLS/TCP connections, and the two endpoints interwork via an MSRP gateway" with "This section describes the network configuration where one MSRP endpoint uses data channels as MSRP transport, the other MSRP endpoint uses TLS/TCP connections as MSRP transport, and the two MSRP endpoints interwork via an MSRP gateway".

10.2. Changes against ’draft-ejzak-mmusic-msrp-usage-data-channel-01’

- Removed empty spaces after ";" in the examples’ "a=dcmap" attribute lines.

- In all examples, the "m" line proto value "DTLS/SCTP" was replaced with "UDP/DTLS/SCTP" and the "a=fmtp" attribute lines were replaced with "a=max-message-size" attribute lines, as per draft-ietf-mmusic-sctp-sdp-12.

10.3. Changes against ’-00’

- Transport parameter change for MSRP to allow MSRP RFC transports.

- Clarification on SDP offer/answer and removing duplicated procedures and refer them to [I-D.ejzak-mmusic-data-channel-sdpneg].

11. References

11.1. Normative References

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

[I-D.ietf-rtcweb-jsep] Uberti, J., Jennings, C., and E. Rescorla, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-08 (work in progress), October 2014.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[I-D.ietf-rtcweb-data-protocol] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", draft-ietf-rtcweb-data-protocol-09 (work in progress), January 2015.

[I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

[I-D.ejzak-mmusic-data-channel-sdpneg] Drage, K., Stoetzer-Bradler, J., Ejzak, R., and J. Marcon, "SDP-based "SCTP over DTLS" data channel negotiation", draft-ejzak-mmusic-data-channel-sdpneg-02 (work in progress), October 2014.

[I-D.ietf-mmusic-data-channel-sdpneg] Drage, K., Stoetzer-Bradler, J., Ejzak, R., Marcon, J., and R. Makaraju, "SDP-based "SCTP over DTLS" data channel negotiation", draft-ietf-mmusic-data-channel-sdpneg-00 (work in progress), January 2015.

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

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for the Message Sessions Relay Protocol (MSRP)", RFC 4976, September 2007.

[RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.

[RFC6135] Holmberg, C. and S. Blau, "An Alternative Connection Model for the Message Session Relay Protocol (MSRP)", RFC 6135, February 2011.

[RFC6714] Holmberg, C., Blau, S., and E. Burger, "Connection Establishment for Media Anchoring (CEMA) for the Message Session Relay Protocol (MSRP)", RFC 6714, August 2012.

11.2. Informative References

[WebRtcAPI] Bergkvist, A., Burnett, D., Narayanan, A., and C. Jennings, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120821, August 2012, <http://www.w3.org/TR/2012/WD-webrtc-20120821>.

Authors' Addresses

Keith Drage (editor)
Alcatel-Lucent
Quadrant, Stonehill Green, Westlea
Swindon
UK

Email: keith.drage@alcatel-lucent.com

Maridi R. Makaraju (Raju)
Alcatel-Lucent
2000 Lucent Lane
Naperville, Illinois
US

Email: Raju.Makaraju@alcatel-lucent.com

Juergen Stoetzer-Bradler
Alcatel-Lucent
Lorenzstrasse 10
D-70435 Stuttgart
Germany

Email: Juergen.Stoetzer-Bradler@alcatel-lucent.com

Richard Ejzak
Unaffiliated

Email: richard.ejzak@gmail.com

Jerome Marcon
Unaffiliated

Network Working Group
Internet-Draft
Obsoletes: 4566 (if approved)
Intended status: Standards Track
Expires: July 25, 2015

M. Handley
UCL
V. Jacobson
PARC
C.S. Perkins
University of Glasgow
A. Begen
Cisco

January 21, 2015

SDP: Session Description Protocol
draft-ietf-mmusic-rfc4566bis-14

Abstract

This memo defines the Session Description Protocol (SDP). SDP is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. This document obsoletes RFC 4566.

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 25, 2015.

Table of Contents

1. Introduction

When initiating multimedia teleconferences, voice-over-IP calls, streaming video, or other sessions, there is a requirement to convey media details, transport addresses, and other session description metadata to the participants.

SDP provides a standard representation for such information, irrespective of how that information is transported. SDP is purely a format for session description -- it does not incorporate a transport protocol, and it is intended to use different transport protocols as appropriate, including the Session Announcement Protocol [RFC2974], Session Initiation Protocol [RFC3261], Real Time Streaming Protocol [RFC2326], electronic mail using the MIME extensions, and the Hypertext Transport Protocol.

SDP is intended to be general purpose so that it can be used in a wide range of network environments and applications. However, it is not intended to support negotiation of session content or media encodings: this is viewed as outside the scope of session description.

This memo obsoletes [RFC4566]. The changes relative to [RFC4566] are limited to essential corrections, and are outlined in Section 10 of this memo.

2. Glossary of Terms

The following term is used in this document and has specific meaning within the context of this document.

Session Description: A well-defined format for conveying sufficient information to discover and participate in a multimedia session.

The terms "multimedia conference" and "multimedia session" are used in this document as defined in [I-D.ietf-avtext-rtp-grouping-taxonomy]. The terms "session" and "multimedia session" are used interchangeably in this document.

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 [RFC2119].

3. Examples of SDP Usage

3.1. Session Initiation

The Session Initiation Protocol (SIP) [RFC3261] is an application-layer control protocol for creating, modifying, and terminating sessions such as Internet multimedia conferences, Internet telephone calls, and multimedia distribution. The SIP messages used to create sessions carry session descriptions that allow participants to agree on a set of compatible media types. These session descriptions are commonly formatted using SDP. When used with SIP, the offer/answer

model [RFC3264] provides a limited framework for negotiation using SDP.

3.2. Streaming Media

The Real Time Streaming Protocol (RTSP) [RFC2326], is an application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. An RTSP client and server negotiate an appropriate set of parameters for media delivery, partially using SDP syntax to describe those parameters.

3.3. Email and the World Wide Web

Alternative means of conveying session descriptions include electronic mail and the World Wide Web (WWW). For both email and WWW distribution, the media type "application/sdp" is used. This enables the automatic launching of applications for participation in the session from the WWW client or mail reader in a standard manner.

Note that announcements of multicast sessions made only via email or the WWW do not have the property that the receiver of a session announcement can necessarily receive the session because the multicast sessions may be restricted in scope, and access to the WWW server or reception of email is possible outside this scope.

3.4. Multicast Session Announcement

In order to assist the advertisement of multicast multimedia conferences and other multicast sessions, and to communicate the relevant session setup information to prospective participants, a distributed session directory may be used. An instance of such a session directory periodically sends packets containing a description of the session to a well-known multicast group. These advertisements are received by other session directories such that potential remote participants can use the session description to start the tools required to participate in the session.

One protocol used to implement such a distributed directory is the Session Announcement Protocol (SAP) [RFC2974]. SDP provides the recommended session description format for such session announcements.

4. Requirements and Recommendations

The purpose of SDP is to convey information about media streams in multimedia sessions to allow the recipients of a session description

to participate in the session. SDP is primarily intended for use in an internetwork, although it is sufficiently general that it can describe multimedia conferences in other network environments. Media streams can be many-to-many. Sessions need not be continually active.

Thus far, multicast-based sessions on the Internet have differed from many other forms of conferencing in that anyone receiving the traffic can join the session (unless the session traffic is encrypted). In such an environment, SDP serves two primary purposes. It is a means to communicate the existence of a session, and it is a means to convey sufficient information to enable joining and participating in the session. In a unicast environment, only the latter purpose is likely to be relevant.

An SDP session description includes the following:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information needed to receive those media (addresses, ports, formats, etc.)

As resources necessary to participate in a session may be limited, some additional information may also be desirable:

- Information about the bandwidth to be used by the session
- Contact information for the person responsible for the session

In general, SDP must convey sufficient information to enable applications to join a session (with the possible exception of encryption keys) and to announce the resources to be used to any non-participants that may need to know. (This latter feature is primarily useful when SDP is used with a multicast session announcement protocol.)

4.1. Media and Transport Information

An SDP session description includes the following media information:

- The type of media (video, audio, etc.)
- The transport protocol (RTP/UDP/IP, H.320, etc.)

- The format of the media (H.261 video, MPEG video, etc.)

In addition to media format and transport protocol, SDP conveys address and port details. For an IP multicast session, these comprise:

- The multicast group address for media
- The transport port for media

This address and port are the destination address and destination port of the multicast stream, whether being sent, received, or both.

For unicast IP sessions, the following are conveyed:

- The remote address for media
- The remote transport port for media

The semantics of this address and port depend on the media and transport protocol defined. By default, this SHOULD be the remote address and remote port to which data is sent. Some media types may redefine this behaviour, but this is NOT RECOMMENDED since it complicates implementations (including middleboxes that must parse the addresses to open Network Address Translation (NAT) or firewall pinholes).

4.2. Timing Information

Sessions may be either bounded or unbounded in time. Whether or not they are bounded, they may be only active at specific times. SDP can convey:

- An arbitrary list of start and stop times bounding the session
- For each bound, repeat times such as "every Wednesday at 10am for one hour"

This timing information is globally consistent, irrespective of local time zone or daylight saving time (see Section 5.9).

4.3. Private Sessions

It is possible to create both public sessions and private sessions. SDP itself does not distinguish between these; private sessions are typically conveyed by encrypting the session description during distribution. The details of how encryption is performed are dependent on the mechanism used to convey SDP; mechanisms are

currently defined for SDP transported using SAP [RFC2974] and SIP [RFC3261], and others may be defined in the future.

If a session announcement is private, it is possible to use that private announcement to convey encryption keys necessary to decode each of the media in a multimedia conference, including enough information to know which encryption scheme is used for each media.

4.4. Obtaining Further Information about a Session

A session description should convey enough information to decide whether or not to participate in a session. SDP may include additional pointers in the form of Uniform Resource Identifiers (URIs) for more information about the session.

4.5. Categorisation

When many session descriptions are being distributed by SAP, or any other advertisement mechanism, it may be desirable to filter session announcements that are of interest from those that are not. SDP supports a categorisation mechanism for sessions that is capable of being automated (the "a=cat:" attribute; see Section 6).

4.6. Internationalisation

The SDP specification recommends the use of the ISO 10646 character set in the UTF-8 encoding [RFC3629] to allow many different languages to be represented. However, to assist in compact representations, SDP also allows other character sets such as ISO 8859-1 to be used when desired. Internationalisation only applies to free-text fields (session name and background information), and not to SDP as a whole.

5. SDP Specification

An SDP session description is denoted by the media type "application/sdp" (See Section 8).

An SDP session description is entirely textual. SDP field names and attribute names use only the US-ASCII subset of UTF-8, but textual fields and attribute values MAY use the full ISO 10646 character set in UTF-8 encoding, or some other character set defined by the "a=charset:" attribute. Field and attribute values that use the full UTF-8 character set are never directly compared, hence there is no requirement for UTF-8 normalisation. The textual form, as opposed to a binary encoding such as ASN.1 or XDR, was chosen to enhance portability, to enable a variety of transports to be used, and to allow flexible, text-based toolkits to be used to generate and process session descriptions. However, since SDP may be used in

environments where the maximum permissible size of a session description is limited, the encoding is deliberately compact. Also, since announcements may be transported via very unreliable means or damaged by an intermediate caching server, the encoding was designed with strict order and formatting rules so that most errors would result in malformed session announcements that could be detected easily and discarded. This also allows rapid discarding of encrypted session announcements for which a receiver does not have the correct key.

An SDP session description consists of a number of lines of text of the form:

<type>=<value>

where <type> MUST be exactly one case-significant character and <value> is structured text whose format depends on <type>. In general, <value> is either a number of fields delimited by a single space character or a free format string, and is case-significant unless a specific field defines otherwise. Whitespace MUST NOT be used on either side of the "=" sign.

An SDP session description consists of a session-level section followed by zero or more media-level sections. The session-level part starts with a "v=" line and continues to the first media-level section (or the end of the whole description, whichever comes first). Each media-level section starts with an "m=" line and continues to the next media-level section or the end of the whole session description - whichever comes first. In general, session-level values are the default for all media unless overridden by an equivalent media-level value.

Some lines in each description are REQUIRED and some are OPTIONAL, but all MUST appear in exactly the order given here (the fixed order greatly enhances error detection and allows for a simple parser). OPTIONAL items are marked with a "*".

```
   Session description
      v=  (protocol version)
      o=  (originator and session identifier)
      s=  (session name)
      i=* (session information)
      u=* (URI of description)
      e=* (email address)
      p=* (phone number)
      c=* (connection information -- not required if included in
           all media descriptions)
```

```
      b=* (zero or more bandwidth information lines)
      One or more time descriptions ("t=" and "r=" lines; see below)
      z=* (time zone adjustments)
      k=* (encryption key)
      a=* (zero or more session attribute lines)
      Zero or more media descriptions

   Time description
      t=  (time the session is active)
      r=* (zero or more repeat times)

   Media description, if present
      m=  (media name and transport address)
      i=* (media title)
      c=* (connection information -- optional if included at
           session level)
      b=* (zero or more bandwidth information lines)
      k=* (encryption key)
      a=* (zero or more media attribute lines)
```

The set of type letters is deliberately small and not intended to be extensible -- an SDP parser MUST completely ignore any session description that contains a type letter that it does not understand. The attribute mechanism ("a=" described below) is the primary means for extending SDP and tailoring it to particular applications or media. Some attributes (the ones listed in Section 6 of this memo) have a defined meaning, but others may be added on an application-, media-, or session-specific basis. An SDP parser MUST ignore any attribute it doesn't understand.

An SDP session description may contain URIs that reference external content in the "u=", "k=", and "a=" lines. These URIs may be dereferenced in some cases, making the session description non-self-contained.

The connection ("c=") information in the session-level section applies to all the media of that session unless overridden by connection information in the media description. For instance, in the example below, each audio media behaves as if it were given a "c=IN IP4 233.252.0.2".

An example SDP description is:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 198.51.100.1
s=SDP Seminar
i=A Seminar on the session description protocol
```

```
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.2
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=audio 49180 RTP/AVP 0
m=video 51372 RTP/AVP 99
c=IN IP4 233.252.0.1/127
a=rtpmap:99 h263-1998/90000
```

Text fields such as the session name and information are octet strings that may contain any octet with the exceptions of 0x00 (Nul), 0x0a (ASCII newline), and 0x0d (ASCII carriage return). The sequence CRLF (0x0d0a) is used to end a record, although parsers SHOULD be tolerant and also accept records terminated with a single newline character. If the "a=charset" attribute is not present, these octet strings MUST be interpreted as containing ISO-10646 characters in UTF-8 encoding (the presence of the "a=charset" attribute may force some fields to be interpreted differently).

A session description can contain domain names in the "o=", "u=", "e=", "c=", and "a=" lines. Any domain name used in SDP MUST comply with [RFC1034], [RFC1035]. Internationalised domain names (IDNs) MUST be represented using the ASCII Compatible Encoding (ACE) form defined in [RFC5890] and MUST NOT be directly represented in UTF-8 or any other encoding (this requirement is for compatibility with [RFC4566] and other early SDP-related standards, which predate the development of internationalised domain names).

5.1. Protocol Version ("v=")

```
v=0
```

The "v=" field gives the version of the Session Description Protocol. This memo defines version 0. There is no minor version number.

5.2. Origin ("o=")

```
o=<username> <sess-id> <sess-version> <nettype> <addrtype>
  <unicast-address>
```

The "o=" field gives the originator of the session (her username and the address of the user's host) plus a session identifier and version number:

<username> is the user's login on the originating host, or it is "-" if the originating host does not support the concept of user IDs. The <username> MUST NOT contain spaces.

<sess-id> is a numeric string such that the tuple of <username>, <sess-id>, <nettype>, <addrtype>, and <unicast-address> forms a globally unique identifier for the session. The method of <sess-id> allocation is up to the creating tool, but it has been suggested that a Network Time Protocol (NTP) format timestamp be used to ensure uniqueness [RFC5905].

<sess-version> is a version number for this session description. Its usage is up to the creating tool, so long as <sess-version> is increased when a modification is made to the session data. Again, it is RECOMMENDED that an NTP format timestamp is used.

<nettype> is a text string giving the type of network. Initially "IN" is defined to have the meaning "Internet", but other values MAY be registered in the future (see Section 8).

<addrtype> is a text string giving the type of the address that follows. Initially "IP4" and "IP6" are defined, but other values MAY be registered in the future (see Section 8).

<unicast-address> is an address of the machine from which the session was created. For an address type of IP4, this is either a fully qualified domain name of the machine or the dotted-decimal representation of an IP version 4 address of the machine. For an address type of IP6, this is either a fully qualified domain name of the machine or the compressed textual representation of an IP version 6 address of the machine. For both IP4 and IP6, the fully qualified domain name is the form that SHOULD be given unless this is unavailable, in which case a globally unique address MAY be substituted. Unless an SDP extension for NAT traversal is used (e.g., ICE [RFC5245], ICE TCP [RFC6544]), a local IP address MUST NOT be used in any context where the SDP description might leave the scope in which the address is meaningful (for example, a local address MUST NOT be included in an application-level referral that might leave the scope).

In general, the "o=" field serves as a globally unique identifier for this version of this session description, and the subfields excepting the version taken together identify the session irrespective of any modifications.

For privacy reasons, it is sometimes desirable to obfuscate the username and IP address of the session originator. If this is a concern, an arbitrary <username> and private <unicast-address> MAY be chosen to populate the "o=" field, provided that these are selected in a manner that does not affect the global uniqueness of the field.

5.3. Session Name ("s=")

s=<session name>

The "s=" field is the textual session name. There MUST be one and only one "s=" field per session description. The "s=" field MUST NOT be empty and SHOULD contain ISO 10646 characters (but see also the "a=charset" attribute). If a session has no meaningful name, the value "s= " SHOULD be used (i.e., a single space as the session name).

5.4. Session Information ("i=")

i=<session description>

The "i=" field provides textual information about the session. There MUST be at most one session-level "i=" field per session description, and at most one "i=" field per media. If the "a=charset" attribute is present, it specifies the character set used in the "i=" field. If the "a=charset" attribute is not present, the "i=" field MUST contain ISO 10646 characters in UTF-8 encoding.

A single "i=" field MAY also be used for each media definition. In media definitions, "i=" fields are primarily intended for labelling media streams. As such, they are most likely to be useful when a single session has more than one distinct media stream of the same media type. An example would be two different whiteboards, one for slides and one for feedback and questions.

The "i=" field is intended to provide a free-form human-readable description of the session or the purpose of a media stream. It is not suitable for parsing by automata.

5.5. URI ("u=")

u=<uri>

A URI is a Uniform Resource Identifier as used by WWW clients [RFC3986]. The URI should be a pointer to additional information

about the session. This field is OPTIONAL, but if it is present it MUST be specified before the first media field. No more than one URI field is allowed per session description.

5.6. Email Address and Phone Number ("e=" and "p=")

```
e=<email-address>
p=<phone-number>
```

The "e=" and "p=" lines specify contact information for the person responsible for the session. This is not necessarily the same person that created the session description.

Inclusion of an email address or phone number is OPTIONAL. Note that the previous version of SDP specified that either an email field or a phone field MUST be specified, but this was widely ignored. The change brings the specification into line with common usage.

If an email address or phone number is present, it MUST be specified before the first media field. More than one email or phone field can be given for a session description.

Phone numbers SHOULD be given in the form of an international public telecommunication number (see ITU-T Recommendation E.164) preceded by a "+". Spaces and hyphens may be used to split up a phone field to aid readability if desired. For example:

```
p=+1 617 555-6011
```

Both email addresses and phone numbers can have an OPTIONAL free text string associated with them, normally giving the name of the person who may be contacted. This MUST be enclosed in parentheses if it is present. For example:

```
e=j.doe@example.com (Jane Doe)
```

The alternative [RFC5322] name quoting convention is also allowed for both email addresses and phone numbers. For example:

```
e=Jane Doe <j.doe@example.com>
```

The free text string SHOULD be in the ISO-10646 character set with UTF-8 encoding, or alternatively in ISO-8859-1 or other encodings if the appropriate session-level "a=charset" attribute is set.

5.7. Connection Data ("c=")

```
c=<nettype> <addrtype> <connection-address>
```

The "c=" field contains connection data.

A session description MUST contain either at least one "c=" field in each media description or a single "c=" field at the session level. It MAY contain a single session-level "c=" field and additional "c=" field(s) per media description, in which case the per-media values override the session-level settings for the respective media.

The first sub-field ("<nettype>") is the network type, which is a text string giving the type of network. Initially, "IN" is defined to have the meaning "Internet", but other values MAY be registered in the future (see Section 8).

The second sub-field ("<addrtype>") is the address type. This allows SDP to be used for sessions that are not IP based. This memo only defines IP4 and IP6, but other values MAY be registered in the future (see Section 8).

The third sub-field ("<connection-address>") is the connection address. OPTIONAL sub-fields MAY be added after the connection address depending on the value of the <addrtype> field.

When the <addrtype> is IP4 and IP6, the connection address is defined as follows:

- If the session is multicast, the connection address will be an IP multicast group address. If the session is not multicast, then the connection address contains the unicast IP address of the expected data source or data relay or data sink as determined by additional attribute fields. It is not expected that unicast addresses will be given in a session description that is communicated by a multicast announcement, though this is not prohibited.

- Sessions using an IP4 multicast connection address MUST also have a time to live (TTL) value present in addition to the multicast address. The TTL and the address together define the scope with which multicast packets sent in this session will be sent. TTL values MUST be in the range 0-255. Although the TTL MUST be specified, its use to scope multicast traffic is deprecated; applications SHOULD use an administratively scoped address instead.

The TTL for the session is appended to the address using a slash as a separator. An example is:

```
c=IN IP4 233.252.0.1/127
```

IP6 multicast does not use TTL scoping, and hence the TTL value MUST NOT be present for IP6 multicast. It is expected that IP6 scoped addresses will be used to limit the scope of multimedia conferences.

Hierarchical or layered encoding schemes are data streams where the encoding from a single media source is split into a number of layers. The receiver can choose the desired quality (and hence bandwidth) by only subscribing to a subset of these layers. Such layered encodings are normally transmitted in multiple multicast groups to allow multicast pruning. This technique keeps unwanted traffic from sites only requiring certain levels of the hierarchy. For applications requiring multiple multicast groups, we allow the following notation to be used for the connection address:

```
<base multicast address>[/<ttl>]/<number of addresses>
```

If the number of addresses is not given, it is assumed to be one. Multicast addresses so assigned are contiguously allocated above the base address, so that, for example:

```
c=IN IP4 233.252.0.1/127/3
```

would state that addresses 233.252.0.1, 233.252.0.2, and 233.252.0.3 are to be used at a TTL of 127. This is semantically identical to including multiple "c=" lines in a media description:

```
c=IN IP4 233.252.0.1/127
c=IN IP4 233.252.0.2/127
c=IN IP4 233.252.0.3/127
```

Similarly, an IP6 example would be:

```
c=IN IP6 FF15::101/3
```

which is semantically equivalent to:

```
c=IN IP6 FF15::101
c=IN IP6 FF15::102
```

c=IN IP6 FF15::103

(remembering that the TTL field is not present in IP6 multicast).

Multiple addresses or "c=" lines MAY be specified on a per-media basis only if they provide multicast addresses for different layers in a hierarchical or layered encoding scheme. They MUST NOT be specified for a session-level "c=" field.

The slash notation for multiple addresses described above MUST NOT be used for IP unicast addresses.

5.8. Bandwidth ("b=")

b=<bwtype>:<bandwidth>

This OPTIONAL field denotes the proposed bandwidth to be used by the session or media. The <bwtype> is an alphanumeric modifier giving the meaning of the <bandwidth> figure. Two values are defined in this specification, but other values MAY be registered in the future (see Section 8 and [RFC3556], [RFC3890]):

CT If the bandwidth of a session or media in a session is different from the bandwidth implicit from the scope, a "b=CT:..." line SHOULD be supplied for the session giving the proposed upper limit to the bandwidth used (the "multimedia conference total" bandwidth). The primary purpose of this is to give an approximate idea as to whether two or more sessions can coexist simultaneously. When using the CT modifier with RTP, if several RTP sessions are part of the multimedia conference, the multimedia conference total refers to total bandwidth of all RTP sessions.

AS The bandwidth is interpreted to be application specific (it will be the application's concept of maximum bandwidth). Normally, this will coincide with what is set on the application's "maximum bandwidth" control if applicable. For RTP-based applications, AS gives the RTP "session bandwidth" as defined in Section 6.2 of [RFC3550].

Note that CT gives a total bandwidth figure for all the media at all sites. AS gives a bandwidth figure for a single media at a single site, although there may be many sites sending simultaneously.

A prefix "X-" is defined for <bwtype> names. This is intended for experimental purposes only. For example:

b=X-YZ:128

Use of the "X-" prefix is NOT RECOMMENDED: instead new modifiers SHOULD be registered with IANA in the standard namespace. SDP parsers MUST ignore bandwidth fields with unknown modifiers. Modifiers MUST be alphanumeric and, although no length limit is given, it is recommended that they be short.

The <bandwidth> is interpreted as kilobits per second by default. The definition of a new <bwtype> modifier MAY specify that the bandwidth is to be interpreted in some alternative unit (the "CT" and "AS" modifiers defined in this memo use the default units).

5.9. Timing ("t=")

t=<start-time> <stop-time>

The "t=" lines specify the start and stop times for a session. Multiple "t=" lines MAY be used if a session is active at multiple irregularly spaced times; each additional "t=" line specifies an additional period of time for which the session will be active. If the session is active at regular times, an "r=" line (see below) should be used in addition to, and following, a "t=" line -- in which case the "t=" line specifies the start and stop times of the repeat sequence.

The first and second sub-fields give the start and stop times, respectively, for the session. These values are the decimal representation of Network Time Protocol (NTP) time values in seconds since 1900 [RFC5905]. To convert these values to UNIX time, subtract decimal 2208988800.

NTP timestamps are elsewhere represented by 64-bit values, which wrap sometime in the year 2036. Since SDP uses an arbitrary length decimal representation, this should not cause an issue (SDP timestamps MUST continue counting seconds since 1900, NTP will use the value modulo the 64-bit limit).

If the <stop-time> is set to zero, then the session is not bounded, though it will not become active until after the <start-time>. If the <start-time> is also zero, the session is regarded as permanent.

User interfaces SHOULD strongly discourage the creation of unbounded and permanent sessions as they give no information about when the session is actually going to terminate, and so make scheduling difficult.

The general assumption may be made, when displaying unbounded sessions that have not timed out to the user, that an unbounded session will only be active until half an hour from the current time or the session start time, whichever is the later. If behaviour other than this is required, an end-time SHOULD be given and modified as appropriate when new information becomes available about when the session should really end.

Permanent sessions may be shown to the user as never being active unless there are associated repeat times that state precisely when the session will be active.

5.10. Repeat Times ("r=")

r=<repeat interval> <active duration> <offsets from start-time>

"r=" fields specify repeat times for a session. For example, if a session is active at 10am on Monday and 11am on Tuesday for one hour each week for three months, then the <start-time> in the corresponding "t=" field would be the NTP representation of 10am on the first Monday, the <repeat interval> would be 1 week, the <active duration> would be 1 hour, and the offsets would be zero and 25 hours. The corresponding "t=" field stop time would be the NTP representation of the end of the last session three months later. By default, all fields are in seconds, so the "r=" and "t=" fields might be the following:

```
t=3034423619 3042462419
r=604800 3600 0 90000
```

To make the description more compact, times may also be given in units of days, hours, or minutes. The syntax for these is a number immediately followed by a single case-sensitive character. Fractional units are not allowed -- a smaller unit should be used instead. The following unit specification characters are allowed:

```
d - days (86400 seconds)
h - hours (3600 seconds)
m - minutes (60 seconds)
s - seconds (allowed for completeness)
```

Thus, the above session announcement could also have been written:

```
r=7d 1h 0 25h
```

Monthly and yearly repeats cannot be directly specified with a single SDP repeat time; instead, separate "t=" fields should be used to explicitly list the session times.

5.11. Time Zones ("z=")

z=<adjustment time> <offset> <adjustment time> <offset> ....

To schedule a repeated session that spans a change from daylight saving time to standard time or vice versa, it is necessary to specify offsets from the base time. This is required because different time zones change time at different times of day, different countries change to or from daylight saving time on different dates, and some countries do not have daylight saving time at all.

Thus, in order to schedule a session that is at the same time winter and summer, it must be possible to specify unambiguously by whose time zone a session is scheduled. To simplify this task for receivers, we allow the sender to specify the NTP time that a time zone adjustment happens and the offset from the time when the session was first scheduled. The "z=" field allows the sender to specify a list of these adjustment times and offsets from the base time.

An example might be the following:

z=2882844526 -1h 2898848070 0

This specifies that at time 2882844526, the time base by which the session's repeat times are calculated is shifted back by 1 hour, and that at time 2898848070, the session's original time base is restored. Adjustments are always relative to the specified start time -- they are not cumulative. Adjustments apply to all "t=" and "r=" lines in a session description.

If a session is likely to last several years, it is expected that the session description will be modified periodically rather than transmit several years' worth of adjustments in one session description.

5.12. Encryption Keys ("k=")

k=<method>
k=<method>:<encryption key>

If transported over a secure and trusted channel, the Session Description Protocol MAY be used to convey encryption keys. A simple mechanism for key exchange is provided by the key field ("k="), although this is primarily supported for compatibility with older implementations and its use is NOT RECOMMENDED. Work is in progress to define new key exchange mechanisms for use with SDP [RFC4567] [RFC4568], and it is expected that new applications will use those mechanisms.

A key field is permitted before the first media entry (in which case it applies to all media in the session), or for each media entry as required. The format of keys and their usage are outside the scope of this document, and the key field provides no way to indicate the encryption algorithm to be used, key type, or other information about the key: this is assumed to be provided by the higher-level protocol using SDP. If there is a need to convey this information within SDP, the extensions mentioned previously SHOULD be used. Many security protocols require two keys: one for confidentiality, another for integrity. This specification does not support transfer of two keys.

The method indicates the mechanism to be used to obtain a usable key by external means, or from the encoded encryption key given. The following methods are defined:

k=clear:<encryption key>

The encryption key is included untransformed in this key field. This method MUST NOT be used unless it can be guaranteed that the SDP is conveyed over a secure channel. The encryption key is interpreted as text according to the charset attribute; use the "k=base64:" method to convey characters that are otherwise prohibited in SDP.

k=base64:<encoded encryption key>

The encryption key is included in this key field but has been base64 encoded [RFC4648] because it includes characters that are prohibited in SDP. This method MUST NOT be used unless it can be guaranteed that the SDP is conveyed over a secure channel.

k=uri:<URI to obtain key>

A Uniform Resource Identifier is included in the key field. The URI refers to the data containing the key, and may require additional authentication before the key can be returned. When a request is made to the given URI, the reply should specify the encoding for the key. The URI is often an Secure Socket

Layer/Transport Layer Security (SSL/TLS)-protected HTTP URI ("https:"), although this is not required.

k=prompt

No key is included in this SDP description, but the session or media stream referred to by this key field is encrypted. The user should be prompted for the key when attempting to join the session, and this user-supplied key should then be used to decrypt the media streams. The use of user-specified keys is NOT RECOMMENDED, since such keys tend to have weak security properties.

The key field MUST NOT be used unless it can be guaranteed that the SDP is conveyed over a secure and trusted channel. An example of such a channel might be SDP embedded inside an S/MIME message or a TLS-protected HTTP session. It is important to ensure that the secure channel is with the party that is authorised to join the session, not an intermediary: if a caching proxy server is used, it is important to ensure that the proxy is either trusted or unable to access the SDP.

5.13. Attributes ("a=")

a=<attribute>
a=<attribute>:<value>

Attributes are the primary means for extending SDP. Attributes may be defined to be used as "session-level" attributes, "media-level" attributes, or both.

A media description may have any number of attributes ("a=" fields) that are media specific. These are referred to as "media-level" attributes and add information about the media stream. Attribute fields can also be added before the first media field; these "session-level" attributes convey additional information that applies to the session as a whole rather than to individual media.

Attribute fields may be of two forms:

- A property attribute is simply of the form "a=<flag>". These are binary attributes, and the presence of the attribute conveys that the attribute is a property of the session. An example might be "a=recvonly".

- o A value attribute is of the form "a=<attribute>:<value>". For example, a whiteboard could have the value attribute "a=orient:landscape"

Attribute interpretation depends on the media tool being invoked. Thus receivers of session descriptions should be configurable in their interpretation of session descriptions in general and of attributes in particular.

Attribute names MUST use the US-ASCII subset of ISO-10646/UTF-8.

Attribute values are octet strings, and MAY use any octet value except 0x00 (Nul), 0x0A (LF), and 0x0D (CR). By default, attribute values are to be interpreted as in ISO-10646 character set with UTF-8 encoding. Unlike other text fields, attribute values are NOT normally affected by the "charset" attribute as this would make comparisons against known values problematic. However, when an attribute is defined, it can be defined to be charset dependent, in which case its value should be interpreted in the session charset rather than in ISO-10646.

Attributes MUST be registered with IANA (see Section 8). If an attribute is received that is not understood, it MUST be ignored by the receiver.

5.14. Media Descriptions ("m=")

m=<media> <port> <proto> <fmt> ...

A session description may contain a number of media descriptions. Each media description starts with an "m=" field and is terminated by either the next "m=" field or by the end of the session description. A media field has several sub-fields:

<media> is the media type. Currently defined media are "audio", "video", "text", "application", and "message", although this list may be extended in the future (see Section 8).

<port> is the transport port to which the media stream is sent. The meaning of the transport port depends on the network being used as specified in the relevant "c=" field, and on the transport protocol defined in the <proto> sub-field of the media field. Other ports used by the media application (such as the RTP Control Protocol (RTCP) port [RFC3550]) MAY be derived algorithmically from the base media port or MAY be specified in a separate attribute (for example, "a=rtcp:" as defined in [RFC3605]).

If non-contiguous ports are used or if they don't follow the parity rule of even RTP ports and odd RTCP ports, the "a=rtcp:" attribute MUST be used. Applications that are requested to send media to a <port> that is odd and where the "a=rtcp:" is present MUST NOT subtract 1 from the RTP port: that is, they MUST send the RTP to the port indicated in <port> and send the RTCP to the port indicated in the "a=rtcp" attribute.

For applications where hierarchically encoded streams are being sent to a unicast address, it may be necessary to specify multiple transport ports. This is done using a similar notation to that used for IP multicast addresses in the "c=" field:

```
m=<media> <port>/<number of ports> <proto> <fmt> ...
```

In such a case, the ports used depend on the transport protocol. For RTP, the default is that only the even-numbered ports are used for data with the corresponding one-higher odd ports used for the RTCP belonging to the RTP session, and the <number of ports> denoting the number of RTP sessions. For example:

```
m=video 49170/2 RTP/AVP 31
```

would specify that ports 49170 and 49171 form one RTP/RTCP pair and 49172 and 49173 form the second RTP/RTCP pair. RTP/AVP is the transport protocol and 31 is the format (see below). If non-contiguous ports are required, they must be signalled using a separate attribute (for example, "a=rtcp:" as defined in [RFC3605]).

If multiple addresses are specified in the "c=" field and multiple ports are specified in the "m=" field, a one-to-one mapping from port to the corresponding address is implied. For example:

```
c=IN IP4 233.252.0.1/127/2
m=video 49170/2 RTP/AVP 31
```

would imply that address 233.252.0.1 is used with ports 49170 and 49171, and address 233.252.0.2 is used with ports 49172 and 49173.

The semantics of multiple "m=" lines using the same transport address are undefined. This implies that, unlike limited past practice, there is no implicit grouping defined by such means and an explicit grouping framework (for example, [RFC5888]) should instead be used to express the intended semantics.

<proto> is the transport protocol. The meaning of the transport protocol is dependent on the address type field in the relevant "c=" field. Thus a "c=" field of IP4 indicates that the transport protocol runs over IP4. The following transport protocols are defined, but may be extended through registration of new protocols with IANA (see Section 8):

* udp: denotes an unspecified protocol running over UDP.

* RTP/AVP: denotes RTP [RFC3550] used under the RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551] running over UDP.

* RTP/SAVP: denotes the Secure Real-time Transport Protocol [RFC3711] running over UDP.

The main reason to specify the transport protocol in addition to the media format is that the same standard media formats may be carried over different transport protocols even when the network protocol is the same -- a historical example is vat Pulse Code Modulation (PCM) audio and RTP PCM audio; another might be TCP/RTP PCM audio. In addition, relays and monitoring tools that are transport-protocol-specific but format-independent are possible.

<fmt> is a media format description. The fourth and any subsequent sub-fields describe the format of the media. The interpretation of the media format depends on the value of the <proto> sub-field.

If the <proto> sub-field is "RTP/AVP" or "RTP/SAVP" the <fmt> sub-fields contain RTP payload type numbers. When a list of payload type numbers is given, this implies that all of these payload formats MAY be used in the session, but the first of these formats SHOULD be used as the default format for the session. For dynamic payload type assignments the "a=rtpmap:" attribute (see Section 6) SHOULD be used to map from an RTP payload type number to a media encoding name that identifies the payload format. The "a=fmtp:" attribute MAY be used to specify format parameters (see Section 6).

If the <proto> sub-field is "udp" the <fmt> sub-fields MUST reference a media type describing the format under the "audio", "video", "text", "application", or "message" top-level media types. The media type registration SHOULD define the packet format for use with UDP transport.

For media using other transport protocols, the <fmt> field is protocol specific. Rules for interpretation of the <fmt> sub-field MUST be defined when registering new protocols (see Section 8.2.2).

Section 3 of [RFC4855] states that the payload format (encoding) names defined in the RTP Profile are commonly shown in upper case, while media subtype names are commonly shown in lower case. It also states that both of these names are case-insensitive in both places, similar to parameter names which are case-insensitive both in media type strings and in the default mapping to the SDP a=fmtp attribute.

6. SDP Attributes

The following attributes are defined. Since application writers may add new attributes as they are required, this list is not exhaustive. Registration procedures for new attributes are defined in Section 8.2.4.

6.1. cat (category)

Name: cat

Value: cat-value

Usage Level: session

Charset Dependent: no

Syntax:

```
cat-value = category
category = non-ws-string
```

Example:

```
a=cat:foo.bar
```

This attribute gives the dot-separated hierarchical category of the session. This is to enable a receiver to filter unwanted sessions by category. There is no central registry of categories. This attribute is obsoleted.

6.2. keywds (keywords)

Name: keywds

Value: keywds-value

Usage Level: session

Charset Dependent: yes

Syntax:

```
keywds-value = keywords
keywords = text
```

Example:

```
a=keywds:SDP session description protocol
```

Like the cat attribute, this is to assist identifying wanted sessions at the receiver. This allows a receiver to select interesting session based on keywords describing the purpose of the session; there is no central registry of keywords. Its value should be interpreted in the charset specified for the session description if one is specified, or by default in ISO 10646/UTF-8. This attribute is obsoleted.

6.3. tool

Name: tool

Value: tool-value

Usage Level: session

Charset Dependent: no

Syntax:

```
tool-value = tool-name-and-version
tool-name-and-version = text
```

Example:

```
a=tool:foobar V3.2
```

This gives the name and version number of the tool used to create the session description.

6.4. ptime (packet time)

Name: ptime

Value: ptime-value

Usage Level: media

Charset Dependent: no

Syntax:

```
ptime-value = packet-time
packet-time = integer
  ; do we want to define a limited range for this?
```

Example:

```
a=ptime:20
```

This gives the length of time in milliseconds represented by the media in a packet. This is probably only meaningful for audio data, but may be used with other media types if it makes sense. It should not be necessary to know ptime to decode RTP or vat audio, and it is intended as a recommendation for the encoding/packetisation of audio.

6.5. maxptime (maximum packet time)

Name: maxptime

Value: maxptime-value

Usage Level: media

Charset Dependent: no

Syntax:

```
maxptime-value = packet-time
```

Example:

```
a=maxptime:20
```

This gives the maximum amount of media that can be encapsulated in each packet, expressed as time in milliseconds. The time SHALL be calculated as the sum of the time the media present in the packet represents. For frame-based codecs, the time SHOULD be an integer multiple of the frame size. This attribute is probably only meaningful for audio data, but may be used with other media types if it makes sense. Note that this attribute was introduced after [RFC2327], and non-updated implementations will ignore this attribute.

6.6. rtpmap

Name: rtpmap

Value: rtpmap-value

Usage Level: media

Charset Dependent: no

Syntax:

```
rtpmap-value = payload-type SP encoding-name
  "/" clock-rate [ "/" encoding-params ]
payload-type = zero-based-integer
encoding-name = token
clock-rate = integer
  ; do we want to define a limited range for this?
encoding-params = channels
  ; 4566 is vague about what this can be. RFC4855 seems to be
  ; the authoritative source, and only allows the
  ; value of the media subtype "channels" parameter - the
  ; number of audio channels.
  ; Does anyone think this can be used for something else???
  ; (The implication that multiple parameters might be included
  ; seems a misdirection - additional parameters are
  ; to go into a=fmtp.)
  ; Does anyone have an example of other parameters
  ; using this field?
channels = integer
  ; Is there any reason to make this less restrictive?
```

This attribute maps from an RTP payload type number (as used in an "m=" line) to an encoding name denoting the payload format to be

used. It also provides information on the clock rate and encoding parameters. Note that the payload type number is indicated in a 7-bit field, limiting the values to incusively between 0 and 127.

Although an RTP profile can make static assignments of payload type numbers to payload formats, it is more common for that assignment to be done dynamically using "a=rtpmap:" attributes. As an example of a static payload type, consider u-law PCM coded single-channel audio sampled at 8 kHz. This is completely defined in the RTP Audio/Video profile as payload type 0, so there is no need for an "a=rtpmap:" attribute, and the media for such a stream sent to UDP port 49232 can be specified as:

```
m=audio 49232 RTP/AVP 0
```

An example of a dynamic payload type is 16-bit linear encoded stereo audio sampled at 16 kHz. If we wish to use the dynamic RTP/AVP payload type 98 for this stream, additional information is required to decode it:

```
m=audio 49232 RTP/AVP 98
a=rtpmap:98 L16/16000/2
```

Up to one rtpmap attribute can be defined for each media format specified. Thus, we might have the following:

```
m=audio 49230 RTP/AVP 96 97 98
a=rtpmap:96 L8/8000
a=rtpmap:97 L16/8000
a=rtpmap:98 L16/11025/2
```

RTP profiles that specify the use of dynamic payload types MUST define the set of valid encoding names and/or a means to register encoding names if that profile is to be used with SDP. The "RTP/AVP" and "RTP/SAVP" profiles use media subtypes for encoding names, under the top-level media type denoted in the "m=" line. In the example above, the media types are "audio/l8" and "audio/l16".

For audio streams, <encoding parameters> indicates the number of audio channels. This parameter is OPTIONAL and may be omitted if the number of channels is one, provided that no additional parameters are needed.

For video streams, no encoding parameters are currently specified.

Additional encoding parameters MAY be defined in the future, but codec-specific parameters SHOULD NOT be added. Parameters added to an "a=rtpmap:" attribute SHOULD only be those required for a session directory to make the choice of appropriate media to participate in a session. Codec-specific parameters should be added in other attributes (for example, "a=fmtp:").

Note: RTP audio formats typically do not include information about the number of samples per packet. If a non-default (as defined in the RTP Audio/Video Profile) packetisation is required, the "ptime" attribute is used as given above.

6.7. Media Direction Attributes

At most one of recvonly/sendrecv/sendonly/inactive MAY appear at session level, and at most one MAY appear in each media section.

If any one of these appears in a media section then it applies for that media section. If none appear in a media section then the one from session level, if any, applies to that media section.

If none of the media direction attributes is present at either session level or media level, "sendrecv" SHOULD be assumed as the default for sessions that are not of the multimedia conference type "broadcast" or "H332" (see below).

Within the following SDP example, the "inactive" attribute applies to audio media and the "recvonly" attribute applies to video media.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 198.51.100.1
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.1/127
t=2873397496 2873404696
a=inactive
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
a=recvonly
```

6.7.1. recvonly (receive-only)

Name: recvonly

Value:

Usage Level: session, media

Charset Dependent: no

Example:

a=recvonly

This specifies that the tools should be started in receive-only mode where applicable. Note that recvonly applies to the media only, not to any associated control protocol (e.g., an RTP-based system in recvonly mode SHOULD still send RTCP packets).

6.7.2. sendrecv (send-receive)

Name: sendrecv

Value:

Usage Level: session, media

Charset Dependent: no

Example:

a=sendrecv

This specifies that the tools should be started in send and receive mode. This is necessary for interactive multimedia conferences with tools that default to receive-only mode.

6.7.3. sendonly (send-only)

Name: sendonly

Value:

Usage Level: session, media

Charset Dependent: no

Example:

```
a=sendonly
```

This specifies that the tools should be started in send-only mode. An example may be where a different unicast address is to be used for a traffic destination than for a traffic source. In such a case, two media descriptions may be used, one sendonly and one recvonly. Note that sendonly applies only to the media, and any associated control protocol (e.g., RTCP) SHOULD still be received and processed as normal.

6.7.4. inactive

Name: inactive

Value:

Usage Level: session, media

Charset Dependent: no

Example:

```
a=inactive
```

This specifies that the tools should be started in inactive mode. This is necessary for interactive multimedia conferences where users can put other users on hold. No media is sent over an inactive media stream. Note that an RTP-based system SHOULD still send RTCP, even if started inactive.

6.8. orient (orientation)

Name: orient

Value: orient-value

Usage Level: media

Charset Dependent: no

Syntax:

```
orient-value = portrait / landscape / seascape
portrait  = %s"portrait"
landscape = %s"landscape"
seascape  = %s"seascape"
   ; NOTE: These names are case-sensitive.
```

Example:

```
a=orient:portrait
```

Normally this is only used for a whiteboard or presentation tool. It specifies the orientation of a the workspace on the screen. Permitted values are "portrait", "landscape", and "seascape" (upside-down landscape).

6.9. type (conference type)

Name: type

Value: type-value

Usage Level: session

Charset Dependent: no

Syntax:

```
type-value = conference-type
conference-type = broadcast / meeting / moderated / test /
H332
broadcast = %s"broadcast"
meeting   = %s"meeting"
moderated = %s"moderated"
test      = %s"test"
H332      = %s"H332"
   ; NOTE: These names are case-sensitive.
```

Example:

```
a=type:moderated
```

This specifies the type of the multimedia conference. Suggested values are "broadcast", "meeting", "moderated", "test", and "H332". "recvonly" should be the default for "type:broadcast" sessions,

"type:meeting" should imply "sendrecv", and "type:moderated" should indicate the use of a floor control tool and that the media tools are started so as to mute new sites joining the multimedia conference.

Specifying the attribute "type:H332" indicates that this loosely coupled session is part of an H.332 session as defined in the ITU H.332 specification [ITU.H332.1998]. Media tools should be started "recvonly".

Specifying the attribute "type:test" is suggested as a hint that, unless explicitly requested otherwise, receivers can safely avoid displaying this session description to users.

6.10. charset (character set)

Name: charset

Value: charset-value

Usage Level: session

Charset Dependent: no

Syntax:

```
charset-value = mime-charset (as defined in <xref
  target="I-D.iana-charset-reg-procedure"/>)
```

This specifies the character set to be used to display the session name and information data. By default, the ISO-10646 character set in UTF-8 encoding is used. If a more compact representation is required, other character sets may be used. For example, the ISO 8859-1 is specified with the following SDP attribute:

```
a=charset:ISO-8859-1
```

The charset specified MUST be one of those registered in the IANA Character Sets registry (http://www.iana.org/assignments/character-sets), such as ISO-8859-1. The character set identifier is a US-ASCII string and MUST be compared against identifiers from the "Name" or "Preferred MIME Name" field of the registry using a case-insensitive comparison. If the identifier is not recognised or not supported, all strings that are affected by it SHOULD be regarded as octet strings.

Note that a character set specified MUST still prohibit the use of bytes 0x00 (Nul), 0x0A (LF), and 0x0d (CR). Character sets requiring the use of these characters MUST define a quoting mechanism that prevents these bytes from appearing within text fields.

6.11. sdplang (SDP language)

Name: sdplang

Value: sdplang-value

Usage Level: session, media

Charset Dependent: no

Syntax:

```
sdplang-value = Language-Tag
; Language-Tag defined in RFC5646
```

Example:

```
a=sdplang:fr
```

This can be a session-level attribute or a media-level attribute. Multiple sdplang attributes can be provided either at session or media level if the session description or media use multiple languages.

As a session-level attribute, it specifies the language for the session description. As a media-level attribute, it specifies the language for any media-level SDP information field associated with that media, overriding any sdplang attributes specified at session-level.

In general, sending session descriptions consisting of multiple languages is discouraged. Instead, multiple descriptions SHOULD be sent describing the session, one in each language. However, this is not possible with all transport mechanisms, and so multiple sdplang attributes are allowed although NOT RECOMMENDED.

The "sdplang" attribute value must be a single [RFC5646] language tag in US-ASCII. An "sdplang" attribute SHOULD be specified when a session is distributed with sufficient scope to cross geographic boundaries, where the language of recipients cannot be assumed, or where the session is in a different language from the locally assumed norm.

6.12. lang (language)

Name: lang

Value: lang-value

Usage Level: session, media

Charset Dependent: no

Syntax:

```
lang-value = Language-Tag
; Language-Tag defined in RFC5646
```

Example:

```
a=lang:de
```

Multiple lang attributes can be provided either at session or media level if the session or media use multiple languages, in which case the order of the attributes indicates the order of importance of the various languages in the session or media, from most important to least important.

As a session-level attribute, it specifies the default language for the session being described. As a media-level attribute, it specifies the language for that media, overriding any session-level languages specified.

The "lang" attribute value must be a single [RFC5646] language tag in US-ASCII. A "lang" attribute SHOULD be specified when a session is of sufficient scope to cross geographic boundaries where the language of recipients cannot be assumed, or where the session is in a different language from the locally assumed norm.

6.13. framerate (frame rate)

Name: framerate

Value: framerate-value

Usage Level: media

Charset Dependent: no

Syntax:

```
framerate-value = positive-real-number
positive-real-number = (integer / "0") [ "." integer ]
  ; Notes:
  ; - this permits a zero value. OK?
  ; - do we want to restrict the range or precision?
```

Example:

```
a=framerate:60
```

This gives the maximum video frame rate in frames/sec. It is intended as a recommendation for the encoding of video data. Decimal representations of fractional values are allowed. It is defined only for video media.

6.14. quality

Name: quality

Value: quality-value

Usage Level: media

Charset Dependent: no

Syntax:

```
quality-value = integer
  ; Do we want to restrict the range?
  ; The definition above limits the range to [0-10]
  ; *for video*, but seems to leave usage open for other media.
```

Example:

```
a=quality:10
```

This gives a suggestion for the quality of the encoding as an integer value. The intention of the quality attribute for video is to specify a non-default trade-off between frame-rate and still-image quality. For video, the value is in the range 0 to 10, with the following suggested meaning:

```
10 - the best still-image quality the compression scheme
     can give.
5  - the default behaviour given no quality suggestion.
0  - the worst still-image quality the codec designer
     thinks is still usable.
```

6.15. fmtp (format parameters)

Name: fmtp

Value: fmtp-value

Usage Level: media

Charset Dependent: no

Syntax:

```
fmtp-value = fmt SP format-specific-params
format-specific-params = byte-string
  ; Notes:
  ; - The format parameters are media type parameters and
 need to reflect their syntax.
```

Example:

```
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
```

This attribute allows parameters that are specific to a particular format to be conveyed in a way that SDP does not have to understand them. The format must be one of the formats specified for the media. Format-specific parameters may be any set of parameters required to be conveyed by SDP and given unchanged to the media tool that will use this format. At most one instance of this attribute is allowed for each format.

7. Security Considerations

SDP is frequently used with the Session Initiation Protocol [RFC3261] using the offer/answer model [RFC3264] to agree on parameters for unicast sessions. When used in this manner, the security considerations of those protocols apply.

SDP is a session description format that describes multimedia sessions. Entities receiving and acting upon an SDP message SHOULD be aware that a session description cannot be trusted unless it has been obtained by an authenticated transport protocol from a known and trusted source. Many different transport protocols may be used to distribute session descriptions, and the nature of the authentication will differ from transport to transport. For some transports, security features are often not deployed. In case a session description has not been obtained in a trusted manner, the endpoint SHOULD exercise care because, among other attacks, the media sessions received may not be the intended ones, the destination where media is sent to may not be the expected one, any of the parameters of the session may be incorrect, or the media security may be compromised. It is up to the endpoint to make a sensible decision taking into account the security risks of the application and the user preferences and may decide to ask the user whether or not to accept the session.

One transport that can be used to distribute session descriptions is the Session Announcement Protocol (SAP). SAP provides both encryption and authentication mechanisms, but due to the nature of session announcements it is likely that there are many occasions where the originator of a session announcement cannot be authenticated because the originator is previously unknown to the receiver of the announcement and because no common public key infrastructure is available.

On receiving a session description over an unauthenticated transport mechanism or from an untrusted party, software parsing the session should take a few precautions. Session descriptions contain information required to start software on the receiver's system. Software that parses a session description MUST NOT be able to start other software except that which is specifically configured as appropriate software to participate in multimedia sessions. It is normally considered inappropriate for software parsing a session description to start, on a user's system, software that is appropriate to participate in multimedia sessions, without the user first being informed that such software will be started and giving the user's consent. Thus, a session description arriving by session announcement, email, session invitation, or WWW page MUST NOT deliver the user into an interactive multimedia session unless the user has explicitly pre-authorised such action. As it is not always simple to tell whether or not a session is interactive, applications that are unsure should assume sessions are interactive.

In this specification, there are no attributes that would allow the recipient of a session description to be informed to start multimedia tools in a mode where they default to transmitting. Under some circumstances it might be appropriate to define such attributes. If this is done, an application parsing a session description containing such attributes SHOULD either ignore them or inform the user that joining this session will result in the automatic transmission of multimedia data. The default behaviour for an unknown attribute is to ignore it.

In certain environments, it has become common for intermediary systems to intercept and analyse session descriptions contained within other signalling protocols. This is done for a range of purposes, including but not limited to opening holes in firewalls to allow media streams to pass, or to mark, prioritize, or block traffic selectively. In some cases, such intermediary systems may modify the session description, for example, to have the contents of the session description match NAT bindings dynamically created. These behaviours are NOT RECOMMENDED unless the session description is conveyed in such a manner that allows the intermediary system to conduct proper checks to establish the authenticity of the session description, and the authority of its source to establish such communication sessions. SDP by itself does not include sufficient information to enable these checks: they depend on the encapsulating protocol (e.g., SIP or RTSP).

Use of the "k=" field poses a significant security risk, since it conveys session encryption keys in the clear. SDP MUST NOT be used to convey key material, unless it can be guaranteed that the channel over which the SDP is delivered is both private and authenticated.

Moreover, the "k=" line provides no way to indicate or negotiate cryptographic key algorithms. As it provides for only a single symmetric key, rather than separate keys for confidentiality and integrity, its utility is severely limited. The use of the "k=" line is NOT RECOMMENDED, as discussed in Section 5.12.

8. IANA Considerations

8.1. The "application/sdp" Media Type

One media type registration from [RFC4566] is to be updated, as defined below.

To: ietf-types@iana.org
Subject: Registration of media type "application/sdp"

Type name: application

Subtype name: sdp

Required parameters: None.

Optional parameters: None.

Encoding considerations:
SDP files are primarily UTF-8 format text. The "a=charset:" attribute may be used to signal the presence of other character sets in certain parts of an SDP file (see Section 6 of RFC XXXX). Arbitrary binary content cannot be directly represented in SDP.

Security considerations:
See Section 7 of RFC XXXX.

Interoperability considerations:
See RFC XXXX.

Published specification:
See RFC XXXX.

Applications which use this media type:
Voice over IP, video teleconferencing, streaming media, instant messaging, among others. See also Section 3 of RFC XXXX.

Additional information:

Magic number(s): None.
File extension(s): The extension ".sdp" is commonly used.

Macintosh File Type Code(s): "sdp "

Person & email address to contact for further information:
IETF MMUSIC working group <mmusic@ietf.org>

Intended usage: COMMON

Author/Change controller:
Authors of RFC XXXX
IETF MMUSIC working group delegated from the IESG

8.2. Registration of Parameters

There are seven field names that are registered with IANA. Using the terminology in the SDP specification Backus-Naur Form (BNF), they are "media", "proto", "fmt", "att-field", "bwtype", "nettype", and "addrtype".

The contact address for all parameters registered below is:

IETF MMUSIC working group <mmusic@ietf.org>

8.2.1. Media Types ("media")

The set of media types is intended to be small and SHOULD NOT be extended except under rare circumstances. The same rules should apply for media names as for top-level media types, and where possible the same name should be registered for SDP as for MIME. For media other than existing top-level media types, a Standards Track RFC MUST be produced for a new top-level media type to be registered, and the registration MUST provide good justification why no existing media name is appropriate (the "Standards Action" policy of [RFC5226].

This memo registers the media types "audio", "video", "text", "application", and "message".

Note: The media types "control" and "data" were listed as valid in an early version of this specification (RFC 2327); however, their semantics were never fully specified and they are not widely used. These media types have been removed in this specification, although they still remain valid media type capabilities for a SIP user agent as defined in [RFC3840]. If these media types are considered useful in the future, a Standards Track RFC MUST be produced to document their use. Until that is done, applications SHOULD NOT use these types and SHOULD NOT declare support for them in SIP capabilities declarations (even though they exist in the registry created by [RFC3840]).

8.2.2. Transport Protocols ("proto")

The "proto" field describes the transport protocol used. This SHOULD reference a standards-track protocol RFC. This memo registers three values: "RTP/AVP" is a reference to [RFC3550] used under the RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551] running over UDP/IP, "RTP/SAVP" is a reference to the Secure Real-time Transport Protocol [RFC3711], and "udp" indicates an unspecified protocol over UDP.

If other RTP profiles are defined in the future, their "proto" name SHOULD be specified in the same manner. For example, an RTP profile whose short name is "XYZ" would be denoted by a "proto" field of "RTP /XYZ".

New transport protocols SHOULD be registered with IANA. Registrations MUST reference an RFC describing the protocol. Such an RFC MAY be Experimental or Informational, although it is preferable that it be Standards Track. Registrations MUST also define the rules by which their "fmt" namespace is managed (see below).

8.2.3. Media Formats ("fmt")

Each transport protocol, defined by the "proto" field, has an associated "fmt" namespace that describes the media formats that may be conveyed by that protocol. Formats cover all the possible encodings that might want to be transported in a multimedia session.

RTP payload formats under the "RTP/AVP" and "RTP/SAVP" profiles MUST use the payload type number as their "fmt" value. If the payload type number is dynamically assigned by this session description, an additional "rtpmap" attribute MUST be included to specify the format name and parameters as defined by the media type registration for the payload format. It is RECOMMENDED that other RTP profiles that are registered (in combination with RTP) as SDP transport protocols specify the same rules for the "fmt" namespace.

For the "udp" protocol, new formats SHOULD be registered. Use of an existing media subtype for the format is encouraged. If no media subtype exists, it is RECOMMENDED that a suitable one be registered through the IETF process [RFC6838] by production of, or reference to, a standards-track RFC that defines the transport protocol for the format.

For other protocols, formats MAY be registered according to the rules of the associated "proto" specification.

Registrations of new formats MUST specify which transport protocols they apply to.

8.2.4. Attribute Names ("att-field")

Attribute field names ("att-field") MUST be registered with IANA and documented, because of noticeable issues due to conflicting attributes under the same name. Unknown attributes in SDP are simply ignored, but conflicting ones that fragment the protocol are a serious problem.

New attribute registrations are accepted according to the "Specification Required" policy of [RFC5226], provided that the specification includes the following information:

- Contact name, email address, and telephone number.
- Attribute name (as it will appear in SDP). This MUST conform to the definition of <att-field>.
- Attribute value: The name of an ABNF syntax rule defining the syntax of the value. Absence of a rule name indicates that the attribute takes no value. Enclosing the rule name in "[" and "]" indicates that a value is optional.
- Usage level of the attribute. (One or more of: session, media, source).
- Whether the attribute value is subject to the charset attribute.
- An ABNF definition of the attribute value rule. The rule MUST NOT match anything that is not also matched by <att-value>. The rule name SHOULD [MUST?] NOT be defined as an Incremental Alternative to <att-value>.
- An explanation of the purpose and usage of the attribute.

- o A specification of appropriate attribute values for this attribute (If not included in syntax).

- o Offer/Answer procedures as explained in [RFC3264].

- o Indication of which "category" [I-D.ietf-mmusic-sdp-mux-attributes] an attribute is associated with.

The above is the minimum that IANA will accept. Attributes that are expected to see widespread use and interoperability SHOULD be documented with a standards-track RFC that specifies the attribute more precisely.

Submitters of registrations should ensure that the specification is in the spirit of SDP attributes, most notably that the attribute is platform independent in the sense that it makes no implicit assumptions about operating systems and does not name specific pieces of software in a manner that might inhibit interoperability.

Submitters of registrations should also carefully choose the attribute usage level. They should not choose only session-level when the attribute can have different values when media is disaggregated, i.e., when each m= section has its own IP address on a different endpoint. In that case the attribute type chosen should be "session, media".

IANA has registered the initial set of attribute names ("att-field" values), with definitions as in Section 6 of this memo (these definitions replace those in [RFC4566]).

8.2.5. Bandwidth Specifiers ("bwtype")

A proliferation of bandwidth specifiers is strongly discouraged.

New bandwidth specifiers ("bwtype" fields) MUST be registered with IANA. The submission MUST reference a standards-track RFC specifying the semantics of the bandwidth specifier precisely, and indicating when it should be used, and why the existing registered bandwidth specifiers do not suffice.

IANA has registered the bandwidth specifiers "CT" and "AS" with definitions as in Section 5.8 of this memo (these definitions update those in [RFC4566]).

8.2.6. Network Types ("nettype")

New network types (the "nettype" field) may be registered with IANA if SDP needs to be used in the context of non-Internet environments. Although these are not normally the preserve of IANA, there may be circumstances when an Internet application needs to interoperate with a non-Internet application, such as when gatewaying an Internet telephone call into the Public Switched Telephone Network (PSTN). The number of network types should be small and should be rarely extended. A new network type cannot be registered without registering at least one address type to be used with that network type. A new network type registration MUST reference an RFC that gives details of the network type and address type and specifies how and when they would be used.

IANA has registered the network type "IN" to represent the Internet, with definition as in Sections 5.2 and 5.7 of this memo (these definitions update those in [RFC4566]).

8.2.7. Address Types ("addrtype")

New address types ("addrtype") may be registered with IANA. An address type is only meaningful in the context of a network type, and any registration of an address type MUST specify a registered network type or be submitted along with a network type registration. A new address type registration MUST reference an RFC giving details of the syntax of the address type. Address types are not expected to be registered frequently.

IANA has registered the address types "IP4" and "IP6" with definitions as in Sections 5.2 and 5.7 of this memo (these definitions update those in [RFC4566]).

8.2.8. Registration Procedure

In the RFC documentation that registers SDP "media", "proto", "fmt", "bwtype", "nettype", and "addrtype" fields, the authors MUST include the following information for IANA to place in the appropriate registry:

- contact name, email address, and telephone number
- name being registered (as it will appear in SDP)
- long-form name in English
- type of name ("media", "proto", "fmt", "bwtype", "nettype", or "addrtype")
- a one-paragraph explanation of the purpose of the registered name

o a reference to the specification for the registered name (this will typically be an RFC number)

IANA may refer any registration to the IESG for review, and may request revisions to be made before a registration will be made.

8.3. Encryption Key Access Methods

The IANA previously maintained a table of SDP encryption key access method ("enckey") names. This table is obsolete, since the "k=" line is not extensible. New registrations MUST NOT be accepted.

9. SDP Grammar

This section provides an Augmented BNF grammar for SDP. ABNF is defined in [RFC5234] and [RFC7405].

```
; SDP Syntax
session-description = proto-version
                      origin-field
                      session-name-field
                      information-field
                      uri-field
                      email-fields
                      phone-fields
                      connection-field
                      bandwidth-fields
                      time-fields
                      key-field
                      attribute-fields
                      media-descriptions

proto-version =       %s"v" "=" 1*DIGIT CRLF
                          ;this memo describes version 0

origin-field =       %s"o" "=" username SP sess-id SP sess-version SP
nettype SP addrtype SP unicast-address CRLF

session-name-field =  %s"s" "=" text CRLF

information-field =   [%s"i" "=" text CRLF]

uri-field =           [%s"u" "=" uri CRLF]

email-fields =        *(%s"e" "=" email-address CRLF)

phone-fields =        *(%s"p" "=" phone-number CRLF)
```

```
   connection-field =    [%s"c" "=" nettype SP addrtype SP
                             connection-address CRLF]
                             ;a connection field must be present
                             ;in every media description or at the
                             ;session-level

   bandwidth-fields =    *(%s"b" "=" bwtype ":" bandwidth CRLF)

   time-fields =         1*( %s"t" "=" start-time SP stop-time
                             *(CRLF repeat-fields) CRLF)
                             [zone-adjustments CRLF]

   repeat-fields =       %s"r" "=" repeat-interval SP typed-time
                             1*(SP typed-time)

   zone-adjustments =    %s"z" "=" time SP ["-"] typed-time
                             *(SP time SP ["-"] typed-time)

   key-field =           [%s"k" "=" key-type CRLF]

   attribute-fields =    *(%s"a" "=" attribute CRLF)

   media-descriptions =  *( media-field
                         information-field
                         *connection-field
                         bandwidth-fields
                         key-field
                         attribute-fields )

   media-field =         %s"m" "=" media SP port ["/" integer]
                             SP proto 1*(SP fmt) CRLF

   ; sub-rules of 'o='
   username =            non-ws-string
                         ;pretty wide definition, but doesn't
                         ;include space

   sess-id =             1*DIGIT
                         ;should be unique for this username/host

   sess-version =        1*DIGIT

   nettype =             token
                         ;typically "IN"

   addrtype =            token
                         ;typically "IP4" or "IP6"
```

```
   ; sub-rules of 'u='
   uri =                 URI-reference
                         ; see RFC 3986

   ; sub-rules of 'e=', see RFC 5322 for definitions
   email-address        = address-and-comment / dispname-and-address
                          / addr-spec
   address-and-comment  = addr-spec 1*SP "(" 1*email-safe ")"
   dispname-and-address = 1*email-safe 1*SP "<" addr-spec ">"

   ; sub-rules of 'p='
   phone-number =        phone *SP "(" 1*email-safe ")" /
                         1*email-safe "<" phone ">" /
                         phone

   phone =               ["+"] DIGIT 1*(SP / "-" / DIGIT)

   ; sub-rules of 'c='
   connection-address =  multicast-address / unicast-address

   ; sub-rules of 'b='
   bwtype =              token

   bandwidth =           1*DIGIT

   ; sub-rules of 't='
   start-time =          time / "0"

   stop-time =           time / "0"

   time =                POS-DIGIT 9*DIGIT
                         ; Decimal representation of NTP time in
                         ; seconds since 1900.  The representation
                         ; of NTP time is an unbounded length field
                         ; containing at least 10 digits.  Unlike the
                         ; 64-bit representation used elsewhere, time
                         ; in SDP does not wrap in the year 2036.

   ; sub-rules of 'r=' and 'z='
   repeat-interval =     POS-DIGIT *DIGIT [fixed-len-time-unit]

   typed-time =          1*DIGIT [fixed-len-time-unit]

   fixed-len-time-unit = %s"d" / %s"h" / %s"m" / %s"s"
   ; NOTE: These units are case-sensitive.

   ; sub-rules of 'k='
   key-type =            %s"prompt"
```

```
                        %s"clear:"
                        %s"base64:"
                        %s"uri:"
                        ; NOTE: These names are case-sensitive.

   base64      =        *base64-unit [base64-pad]
   base64-unit =        4base64-char
   base64-pad  =        2base64-char "==" / 3base64-char "="
   base64-char =        ALPHA / DIGIT / "+" / "/"

   ; sub-rules of 'a='
   attribute =          (att-field ":" att-value) / att-field

   att-field =          token

   att-value =          byte-string

   ; sub-rules of 'm='
   media =              token
                        ;typically "audio", "video", "text", or
                        ;"application"

   fmt =                token
                        ;typically an RTP payload type for audio
                        ;and video media

   proto  =             token *("/" token)
                        ;typically "RTP/AVP" or "udp"

   port =               1*DIGIT

   ; generic sub-rules: addressing
   unicast-address =    IP4-address / IP6-address / FQDN / extn-addr

   multicast-address =  IP4-multicast / IP6-multicast / FQDN
                        / extn-addr

   IP4-multicast =      m1 3( "." decimal-uchar )
                        "/" ttl [ "/" integer ]
                        ; IP4 multicast addresses may be in the
                        ; range 224.0.0.0 to 239.255.255.255

   m1 =                 ("22" ("4"/"5"/"6"/"7"/"8"/"9")) /
                        ("23" DIGIT )

   IP6-multicast =      IP6-address [ "/" integer ]
                        ; IP6 address starting with FF
```

```
   ttl =                (POS-DIGIT *2DIGIT) / "0"

   FQDN =               4*(alpha-numeric / "-" / ".")
                        ; fully qualified domain name as specified
                        ; in RFC 1035 (and updates)

   IP4-address =        b1 3("." decimal-uchar)

   b1 =                 decimal-uchar
                        ; less than "224"

   IP6-address =                                    6( h16 ":" ) ls32
                        /                      "::" 5( h16 ":" ) ls32
                        / [               h16 ] "::" 4( h16 ":" ) ls32
                        / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                        / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                        / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                        / [ *4( h16 ":" ) h16 ] "::"              ls32
                        / [ *5( h16 ":" ) h16 ] "::"              h16
                        / [ *6( h16 ":" ) h16 ] "::"

   h16 =                1*4HEXDIG

   ls32 =               ( h16 ":" h16 ) / IP4-address

   ; Generic for other address families
   extn-addr =      non-ws-string

   ; generic sub-rules: datatypes
   text =               byte-string
                        ;default is to interpret this as UTF8 text.
                        ;ISO 8859-1 requires "a=charset:ISO-8859-1"
                        ;session-level attribute to be used

   byte-string =        1*(%x01-09/%x0B-0C/%x0E-FF)
                        ;any byte except NUL, CR, or LF

   non-ws-string =      1*(VCHAR/%x80-FF)
                        ;string of visible characters

   token-char =         ALPHA / DIGIT
                            / "!" / "#" / "$" / "%" / "&" /
                            / "'" ; (single quote)
                            / "*" / "+" / "-" / "." / "^" / "_"
                            / "`" ; (Grave accent)
                            / "{" / "|" / "}" / "~"

   zero-based-integer = "0" / integer
```

```
token =              1*(token-char)

email-safe =         %x01-09/%x0B-0C/%x0E-27/%x2A-3B/%x3D/%x3F-FF
                     ;any byte except NUL, CR, LF, or the quoting
                     ;characters ()<>

integer =            POS-DIGIT *DIGIT

; generic sub-rules: primitives
alpha-numeric =      ALPHA / DIGIT

POS-DIGIT =          %x31-39 ; 1 - 9

decimal-uchar =      DIGIT
                     / POS-DIGIT DIGIT
                     / ("1" 2*(DIGIT))
                     / ("2" ("0"/"1"/"2"/"3"/"4") DIGIT)
                     / ("2" "5" ("0"/"1"/"2"/"3"/"4"/"5"))

; external references:
; ALPHA, DIGIT, CRLF, SP, VCHAR: from RFC 5234
; URI-reference: from RFC 3986
; addr-spec: from RFC 5322
```

10. Summary of Changes from RFC 4566

The ABNF rule for IP6-address has been corrected. As a result, the ABNF rule for IP6-multicast has changed, and the (now unused) rules for hexpart, hexseq, and hex4 have been removed.

IP4 unicast and multicast addresses in the example SDP descriptions have been revised per RFCs 5735 and 5771.

Text in Section 5.2 has been revised to clarify the use of local addresses in case of ICE-like SDP extensions.

Normative and informative references have been updated.

The text regarding the session vs. media-level attribute usage has been clarified.

The case-insensitivity rules from RFC 4855 have been included in this document.

11. Acknowledgements

Many people in the IETF Multiparty Multimedia Session Control (MMUSIC) working group have made comments and suggestions contributing to this document.

12. References

12.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

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

[I-D.ietf-avtext-rtp-grouping-taxonomy]

Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-02 (work in progress), June 2014.

[I-D.iana-charset-reg-procedure]
McFadden, M. and A. Melnikov, "IANA Charset Registration Procedures", draft-iana-charset-reg-procedure-00 (work in progress), October 2014.

[I-D.ietf-mmusic-sdp-mux-attributes]
Nandakumar, S., "A Framework for SDP Attributes when Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-08 (work in progress), January 2015.

12.2. Informative References

[RFC2327] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[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.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[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.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[RFC3890] Westerlund, M., "A Transport Independent Bandwidth Modifier for the Session Description Protocol (SDP)", RFC 3890, September 2004.

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

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B.B., and A.B. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, December 2014.

[ITU.H332.1998]
International Telecommunication Union, "H.323 extended for loosely coupled conferences", ITU Recommendation H.332, September 1998.

[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.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

[RFC2365] Meyer, D., "Administratively Scoped IP Multicast", BCP 23, RFC 2365, July 1998.

Authors' Addresses

Mark Handley
University College London
Department of Computer Science
London WC1E 6BT
UK

EMail: M.Handley@cs.ucl.ac.uk

Van Jacobson
PARC
3333 Coyote Hill Road
Palo Alto, CA 94304
USA

EMail: van@parc.com

Colin Perkins
University of Glasgow
School of Computing Science
University of Glasgow
Glasgow G12 8QQ
UK

EMail: csp@csperkins.org

Ali Begen
Cisco
181 Bay Street
Toronto, ON M5J 2T3
Canada

EMail: abegen@cisco.com

MMUSIC
Internet-Draft
Obsoletes: 5245 (if approved)
Intended status: Standards Track
Expires: September 10, 2015

A. Keranen
Ericsson
J. Rosenberg
jdrosen.net

March 9, 2015

Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols
draft-ietf-mmusic-rfc5245bis-04

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 September 10, 2015.

Copyright Notice

Table of Contents

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.

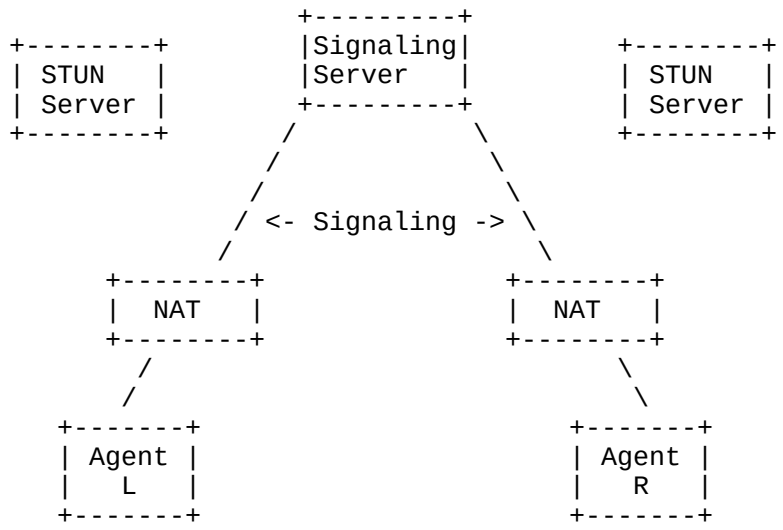


Figure 1: ICE Deployment Scenario

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:

- A transport address on a directly attached network interface
- A translated transport address on the public side of a NAT (a "server reflexive" address)
- 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.

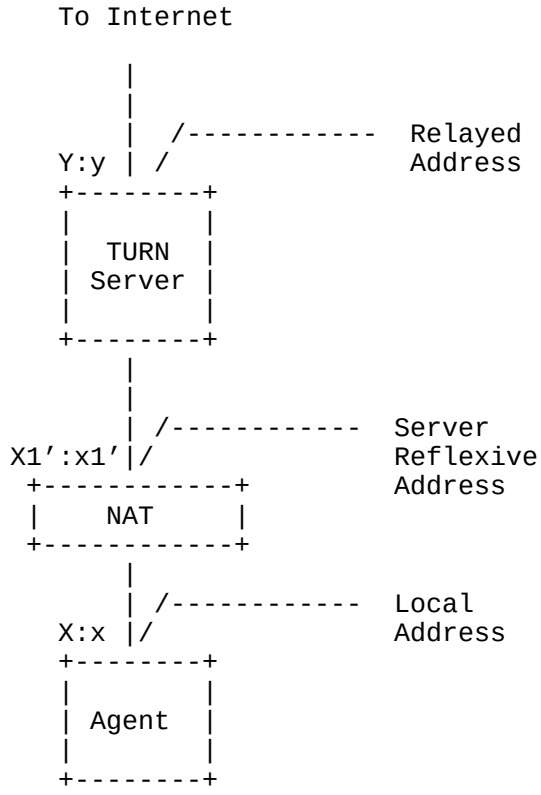


Figure 2: Candidate Relationships

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 X1':x1', 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, X1':x1' 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 X1':x1', 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 X1':x1' 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:

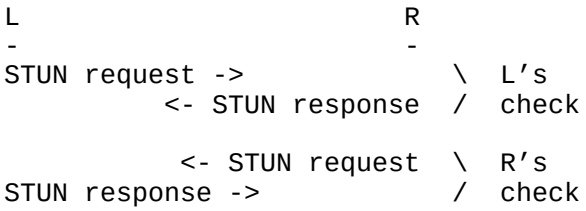


Figure 3: Basic Connectivity Check

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:

- Each agent gives its candidates a numeric priority, which is sent along with the candidate to the peer.

- 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.

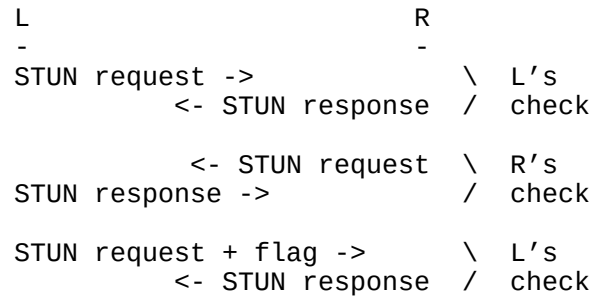


Figure 4: Regular Nomination

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.

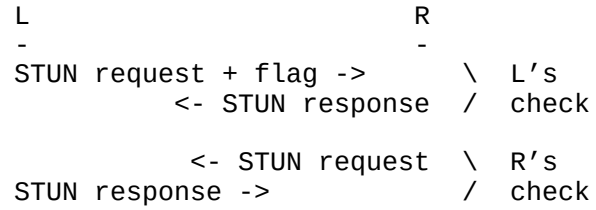


Figure 5: Aggressive Nomination

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.ietf-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: An agent is the protocol implementation involved in the offer/answer exchange. There are two agents involved in an offer/answer exchange.

ICE offer/answer: The process where the ICE agents exchange information (e.g., candidates and passwords) that is needed to perform ICE. RFC 3264 offer/answer with SDP is one example of a protocol that can be used for ICE offer and answer.

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, Media Session: 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", "media stream", and "media session" are still used in this specification to refer to the IP data packets that are exchanged between the peers on the path created and tested with ICE.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. Sending the Initial Offer

In order to send the initial offer in an offer/answer exchange, an agent must (1) gather candidates, (2) prioritize them, (3) eliminate redundant candidates, (4) (possibly) choose default candidates, and then (5) formulate and send the offer. All but the last of these five steps differ for full and lite implementations.

4.1. Full Implementation Requirements

4.1.1. Gathering Candidates

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

4.1.1.1. Host Candidates

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

For each UDP media stream the agent wishes to use, the agent SHOULD obtain a candidate for each component of the media stream on each IP address that the host has, with the exceptions listed below. The agent obtains each candidate by binding to a UDP port on the specific IP address. A host candidate (and indeed every candidate) is always associated with a specific component for which it is a candidate.

Each component has an ID assigned to it, called the component ID. For RTP-based media streams, the RTP itself has a component ID of 1, and RTCP a component ID of 2. If an agent is using RTCP, it MUST obtain a candidate for it. If an agent is using both RTP and RTCP, it would end up with 2*K host candidates if an agent has K IP addresses.

For other than RTP-based streams, use of multiple components is discouraged since using them increases the complexity of ICE

processing. If multiple components are needed, the component IDs SHOULD start with 1 and increase by 1 for each component.

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

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

- Addresses from a loopback interface MUST NOT be included in the candidate addresses.
- Deprecated IPv4-compatible IPv6 addresses [RFC4291] and IPv6 site-local unicast addresses [RFC3879] MUST NOT be included in the address candidates.
- IPv4-mapped IPv6 addresses SHOULD NOT be included in the offered candidates unless the application using ICE does not support IPv4 (i.e., is an IPv6-only application [RFC4038]).
- If one or more host candidates corresponding to an IPv6 address generated using a mechanism that prevents location tracking [I-D.ietf-6man-ipv6-address-generation-privacy] are gathered, host candidates corresponding to IPv6 addresses that do allow location tracking, that are configured on the same interface, and are part of the same network prefix MUST NOT be gathered; and host candidates corresponding to IPv6 link-local addresses MUST NOT be gathered.

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 Ta 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 Ta milliseconds. See Section 13 for guidance on how to set Ta 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:

- they are of the same type (host, relayed, server reflexive, or peer reflexive)
- their bases have the same IP address (the ports can be different)
- for reflexive and relayed candidates, the STUN or TURN servers used to obtain them have the same IP address
- 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:

```
priority = (2^24)*(type preference) +
           (2^8)*(local preference) +
           (2^0)*(256 - 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. 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 or if an agent is using multiple TURN servers. If a host is multihomed because it is dual-stack, the local preference SHOULD be set equal to the precedence value for IP addresses described in RFC 6724 [RFC6724]. If the host operating system provides an API for discovering preference among different addresses, those preferences SHOULD be used for the local preference to prioritize addresses indicated as preferred by the operating system.

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

4.1.2.2. Guidelines for Choosing Type and Local Preferences

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

Furthermore, if an agent is multihomed and has multiple IP addresses, the local preference for host candidates from a VPN interface SHOULD have a priority of 0. If multiple TURN servers are used, local priorities for the candidates obtained from the TURN servers are chosen in a similar fashion as for multihomed local candidates: the local preference value is used to indicate preference among different servers but the preference MUST be unique for each one.

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:

```
priority = (2^24)*(126) +
           (2^8)*(IP precedence) +
           (2^0)*(256 - 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. These MAY be omitted or set to invalid values if the agent does not want to reveal them, e.g., for privacy reasons.

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.

Connectivity check pacing value: If an agent wants to use other than the default pacing values for the connectivity checks, it MUST indicate this in the ICE exchange.

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 11.1, 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 15.4.1, agents MAY limit the number of candidates they'll accept in an offer or answer. A local candidate is paired with a remote candidate if and only if the two candidates have the same component ID and have the same IP address version. It is possible that some of the local candidates won't get paired with remote candidates, and some of the remote candidates won't get paired with local candidates. This can happen if one agent doesn't include candidates for the all of the components for a media stream. If this happens, the number of components for that media stream is effectively reduced, and considered to be equal to the minimum across both agents of the maximum component ID provided by each agent across all components for the media stream.

In the case of RTP, this would happen when one agent provides candidates for RTCP, and the other does not. As another example, the offerer can multiplex RTP and RTCP on the same port and signals that

it can do that in the SDP through an SDP attribute [RFC5761]. However, since the offerer doesn't know if the answerer can perform such multiplexing, the offerer includes candidates for RTP and RTCP on separate ports, so that the offer has two components per media stream. If the answerer can perform such multiplexing, it would include just a single component for each candidate -- for the combined RTP/RTCP mux. ICE would end up acting as if there was just a single component for this candidate.

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

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

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

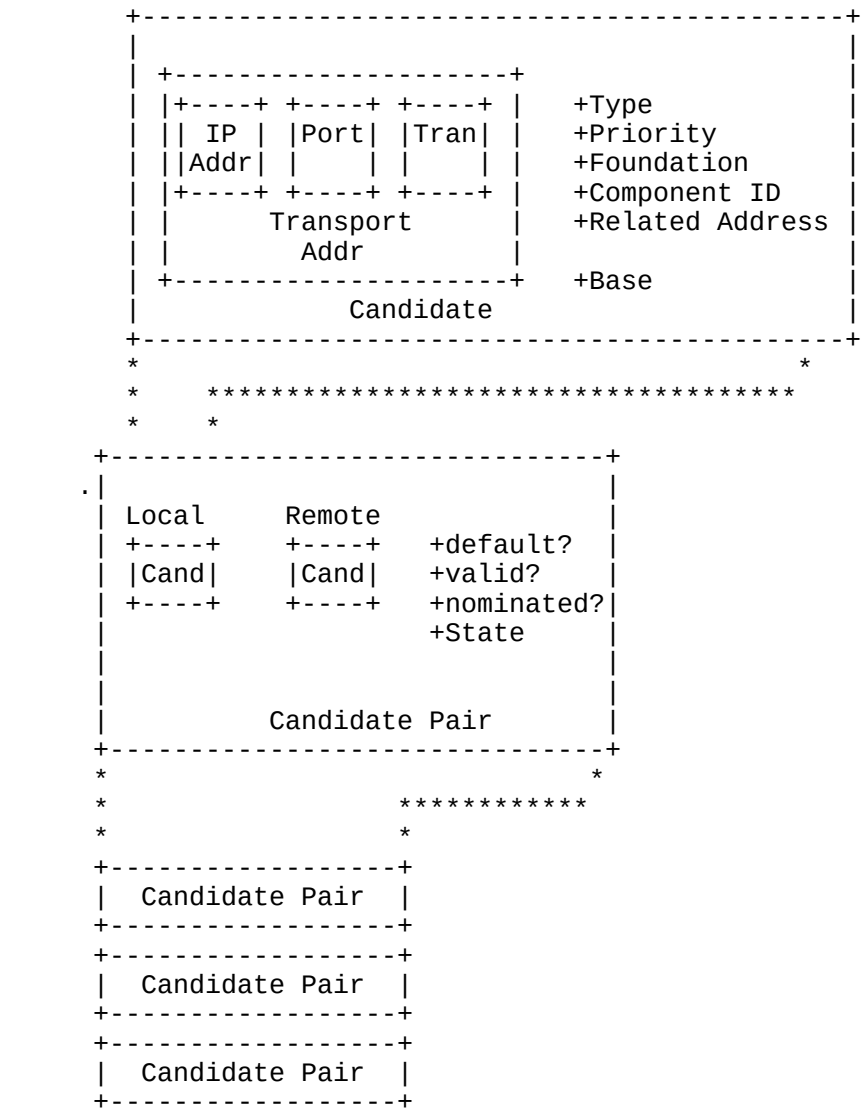


Figure 6: Conceptual Diagram of a Check List

5.6.2. Computing Pair Priority and Ordering Pairs

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

```
pair priority = 2^32*MIN(G,D) + 2*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 15.4.1, 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.

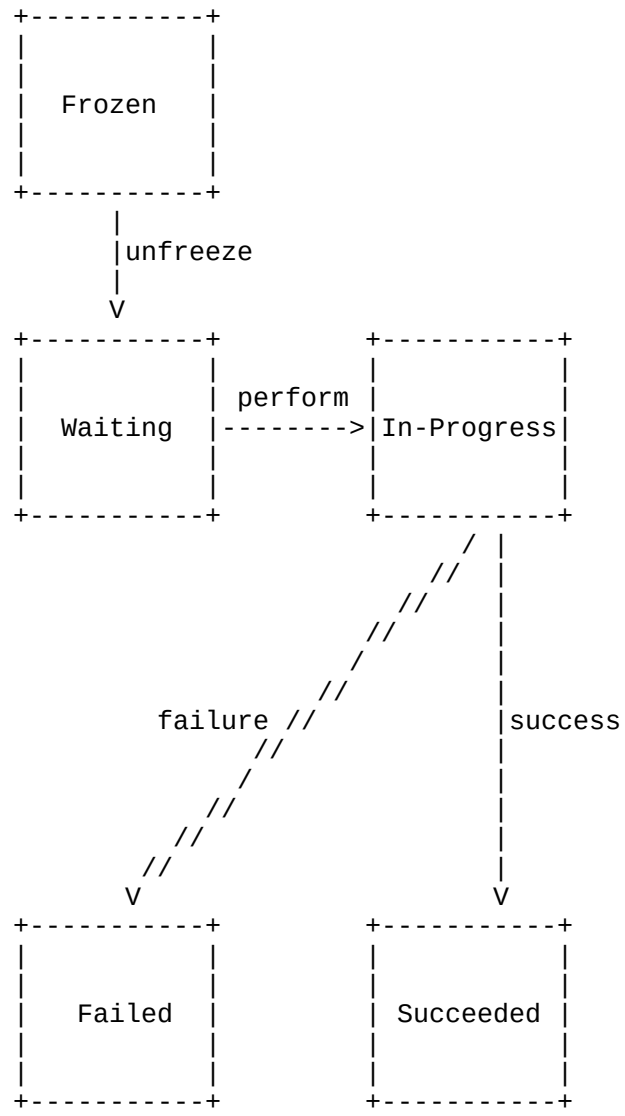


Figure 7: Pair State FSM

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 Ta*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. Ta and the retransmit timer RTO are computed as described in Section 13. 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 Ta 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:

- Find the highest-priority pair in that check list that is in the Waiting state.
- 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.
- 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 16.1, 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 16.1.

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:

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

- 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:

- Its type is equal to peer reflexive.
- Its base is set equal to the local candidate of the candidate pair from which the STUN check was sent.
- Its priority is set equal to the value of the PRIORITY attribute in the Binding request.
- 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:

- 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.

- For each frozen check list, the agent

  * groups together all of the pairs with the same foundation, and

  * for each group, sets the state of the pair with the lowest component ID to Waiting. If there is more than one such pair, the one with the highest-priority is used.

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

7.2. STUN Server Procedures

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

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

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

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

to Binding requests. The same would apply to any layer 2 markings the endpoint might be applying to media packets.

7.2.1. Additional Procedures for Full Implementations

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

7.2.1.1. Detecting and Repairing Role Conflicts

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

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

- 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.

- 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.

- 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.

- 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:

- 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.

- 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:

- 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.
- 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 12). 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.

If certain connectivity check messages are lost, ICE agents using aggressive nomination may end up with different views on the selected candidate pair. In this case, if a security protocol that is able to authenticate the communicating parties (e.g., DTLS) is used, the controlled agent may receive valid secured traffic or handshake initialization originating from the controlling agent on a candidate pair that is different from the one the controlled agent considers as the selected pair. If this happens, the controlled agent MUST consider the pair with the secured traffic as the correct selected pair. If such security protocol is not used, both agents SHOULD continue sending connectivity check messages on the selected pair even after a pair has already been selected for use. In order to prevent the problem described here, at least one check from both agents needs to fully succeed on the selected pair.

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:

- 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.

- 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.
- 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 11.1.
- 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.
- 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 11.1, 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. 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:

- 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.

- 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.

To restart ICE, an agent MUST change both the password and the user name fragment for the media stream in an offer. The set of candidates in the new offer MAY include some, none, or all of the previous candidates for that stream and MAY include a totally new set of candidates

10. Keepalives

All endpoints MUST send keepalives for each media session. These keepalives serve the purpose of keeping NAT bindings alive for the media session. These keepalives MUST be sent even if ICE is not being utilized for the session at all. The keepalive SHOULD be sent using a format that is supported by its peer. ICE endpoints allow for STUN-based keepalives for UDP streams, and as such, STUN keepalives MUST be used when an agent is a full ICE implementation and is communicating with a peer that supports ICE (lite or full). If the peer does not support ICE, the choice of a packet format for keepalives is a matter of local implementation. A format that allows packets to easily be sent in the absence of actual media content is RECOMMENDED. Examples of formats that readily meet this goal are RTP No-Op [I-D.ietf-avt-rtp-no-op], and in cases where both sides support it, RTP comfort noise [RFC3389]. If the peer doesn’t support any formats that are particularly well suited for keepalives, an agent SHOULD send RTP packets with an incorrect version number, or some other form of error that would cause them to be discarded by the peer.

If there has been no packet sent on the candidate pair ICE is using for a media component for Tr seconds (where packets include those defined for the component (RTP or RTCP) and previous keepalives), an agent MUST generate a keepalive on that pair. Tr SHOULD be configurable and SHOULD have a default of 15 seconds. Tr MUST NOT be

configured to less than 15 seconds. Alternatively, if an agent has a dynamic way to discover the binding lifetimes of the intervening NATs, it can use that value to determine Tr. Administrators deploying ICE in more controlled networking environments SHOULD set Tr to the longest duration possible in their environment.

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

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

11. Media Handling

11.1. Sending Media

Procedures for sending media differ for full and lite implementations.

11.1.1. 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:

- 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

- 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

- 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.

11.1.2. 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.

11.1.3. 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.

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. 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.

13. 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 13.1 MAY be followed to rate-control the ICE exchanges. For all other streams, the computation in Section 13.2 MUST be followed.

13.1. Real-time 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:

$$Ta_i = (stun\_packet\_size / rtp\_packet\_size) * rtp\_ptime$$

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

where k is the number of media streams. During the gathering phase, Ta is computed based on the number of media streams the agent has indicated in its offer or answer, and the RTP packet size and RTP ptime are those of the most preferred codec for each media stream.

Once an offer and answer have been exchanged, the agent recomputes Ta to pace the connectivity checks. In that case, the value of Ta is based on the number of media streams that will actually be used in the session, and the RTP packet size and RTP ptime are those of the most preferred codec with which the agent will send.

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

```
RTO = MAX (100ms, Ta * (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 = MAX (100ms, Ta*N * (Num-Waiting + 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.

13.2. Non-real-time Sessions

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

If other Ta value than the default is used, the agent MUST indicate the value it prefers to use in the ICE exchange. Both agents MUST use the higher out of the two proposed values.

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

```
RTO = MAX (500ms, Ta * (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 = MAX (500ms, Ta*N * (Num-Waiting + Num-In-Progress))
```

14. Example

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

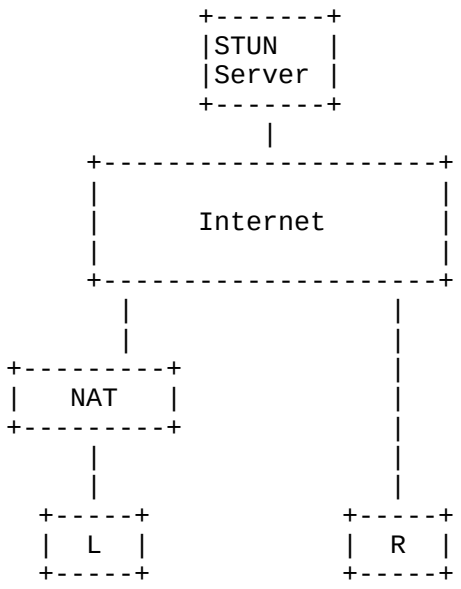


Figure 8: Example Topology

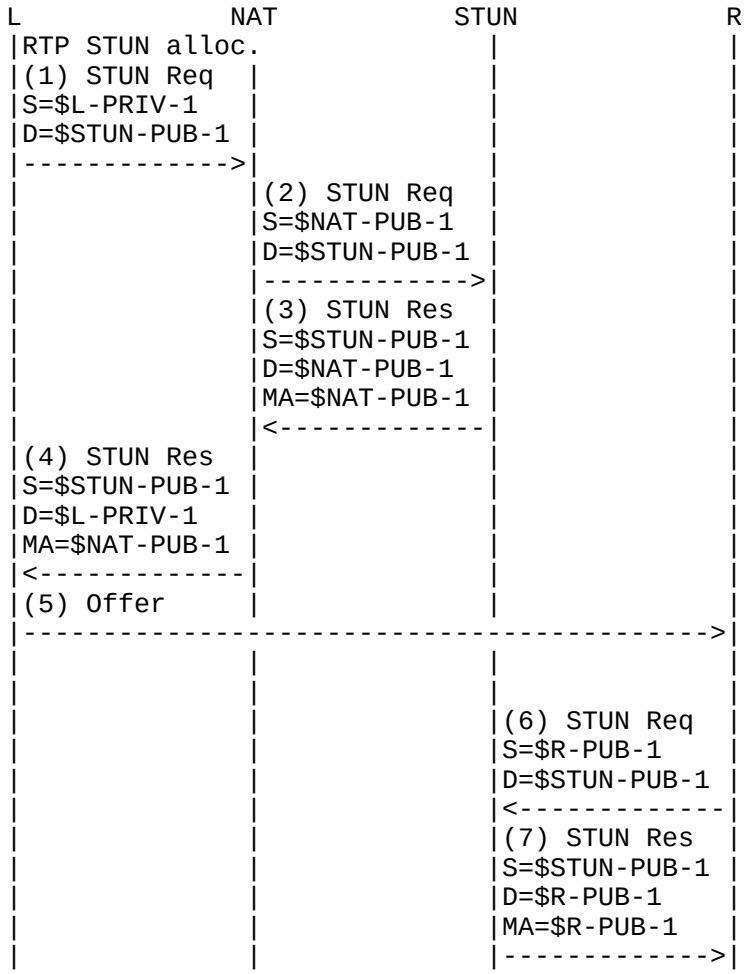
Two agents, L and R, are using ICE. Both are full-mode ICE implementations and use aggressive nomination when they are controlling. Both agents have a single IPv4 address. For agent L, it is 10.0.1.1 in private address space [RFC1918], and for agent R, 192.0.2.1 on the public Internet. Both are configured 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.



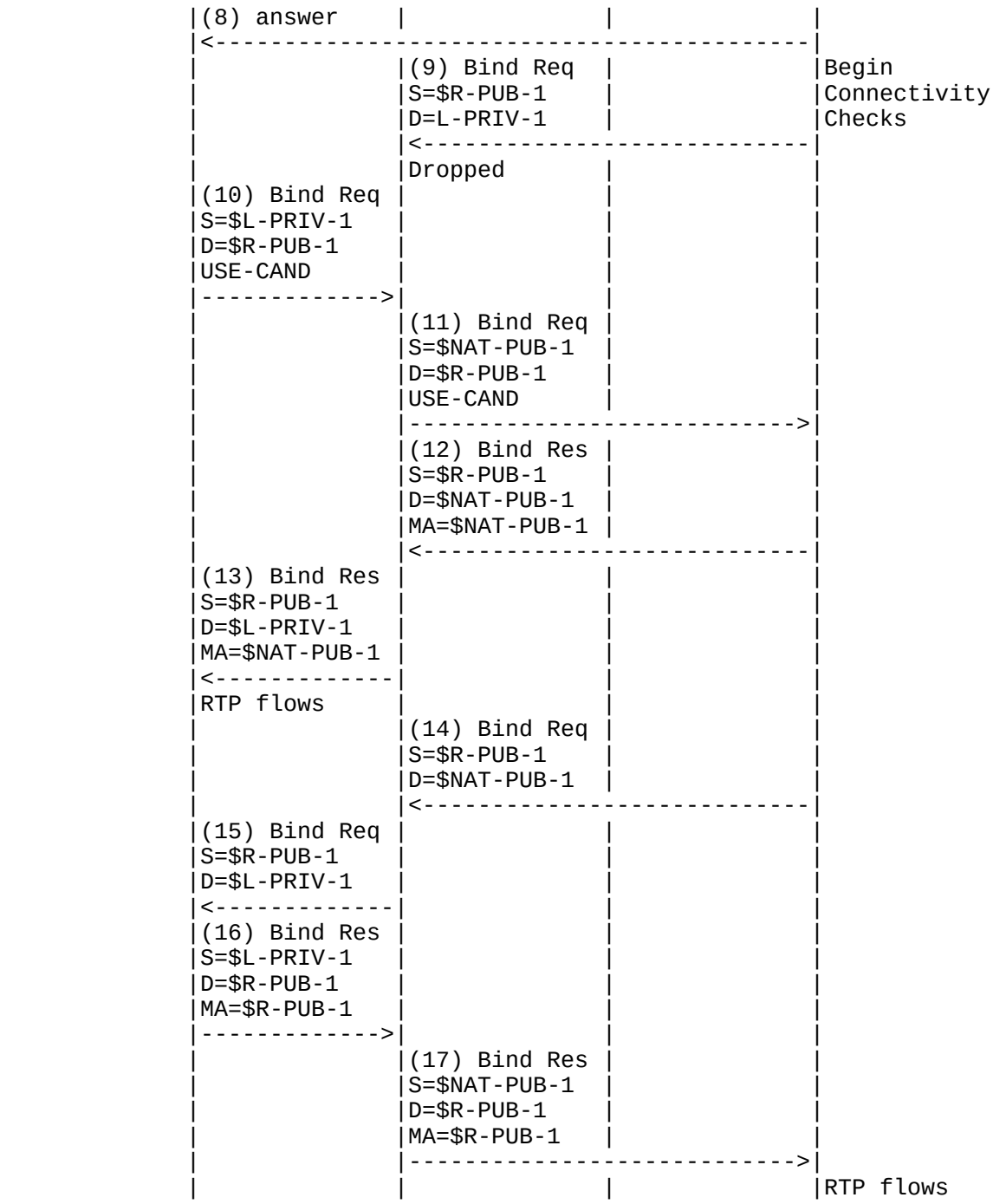

(8) answer
(9) Bind Req
S=$R-PUB-1
D=L-PRIV-1
Begin
Connectivity
Checks
Dropped
(10) Bind Req
S=$L-PRIV-1
D=$R-PUB-1
USE-CAND
(11) Bind Req
S=$NAT-PUB-1
D=$R-PUB-1
USE-CAND
(12) Bind Res
S=$R-PUB-1
D=$NAT-PUB-1
MA=$NAT-PUB-1
(13) Bind Res
S=$R-PUB-1
D=$L-PRIV-1
MA=$NAT-PUB-1
RTP flows
(14) Bind Req
S=$R-PUB-1
D=$NAT-PUB-1
(15) Bind Req
S=$R-PUB-1
D=$L-PRIV-1
(16) Bind Res
S=$L-PRIV-1
D=$R-PUB-1
MA=$R-PUB-1
(17) Bind Res
S=$NAT-PUB-1
D=$R-PUB-1
MA=$R-PUB-1
RTP flows

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.

15. Security Considerations

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

- Using ICE in conjunction with secure signaling techniques, such as SIPS.

- Limiting the total number of connectivity checks to 100, and optionally limiting the number of candidates they'll accept in an offer or answer.

15.1. 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.

15.2. 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:

- 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.

- 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.

- 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 themself.

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.

15.3. 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.

15.4. 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.

15.4.1. 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 Ta ms (say, Ta=20ms). 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 Ta 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:

- There was no response because the initiator is being used to launch a DoS attack against an unsuspecting target that will not respond.
- 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.

16. STUN Extensions

16.1. 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.

16.2. 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.

17. Operational Considerations

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

17.1. 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.

17.2. Bandwidth Requirements

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

17.2.1. 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*1.7bps. 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*3.2 kbps. For a population of one million users, this is 3.2 Gbps, assuming a 10% usage of TURN servers.

17.2.2. 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.

17.2.3. 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.

17.3. 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.

17.4. 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.

17.5. 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.

18. 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.

18.1. STUN Attributes

IANA has registered four STUN attributes:

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

18.2. 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.

19. 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.

19.1. 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.

19.2. Exit Strategy

>From RFC 3424, any UNSAF proposal must provide:

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

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

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

19.3. 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.

19.4. 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.

19.5. 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.

20. Changes from RFC 5245

Following is the list of changes from RFC 5245

- 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.ietf-mmusic-ice-sip-sdp].

- 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.

- With IPv6, the precedence rules of RFC 6724 are used instead of the obsoleted RFC 3483 and using address preferences provided by the host operating system is recommended.

- Candidate gathering rules regarding loopback addresses and IPv6 addresses were clarified.

21. 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. For additional contributions

to this revision of the specification we would like to thank Christer Holmberg, Emil Ivov, Paul Kyzivat, Pal-Erik Martinsen, Simon Perrault, Eric Rescorla, Thomas Stach, Peter Thatcher, Martin Thomson, and Justin Uberti.

22. References

22.1. 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.

22.2. 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.

[RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, September 2004.

[RFC4038] Shin, M-K., Hong, Y-G., Hagino, J., Savola, P., and E. Castro, "Application Aspects of IPv6 Transition", RFC 4038, March 2005.

[RFC4091] Camarillo, G. and J. Rosenberg, "The Alternative Network Address Types (ANAT) Semantics for the Session Description Protocol (SDP) Grouping Framework", RFC 4091, June 2005.

[RFC4092] Camarillo, G. and J. Rosenberg, "Usage of the Session Description Protocol (SDP) Alternative Network Address Types (ANAT) Semantics in the Session Initiation Protocol (SIP)", RFC 4092, June 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

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

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC1918] Rekhter, Y., Moskowitz, 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.

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

[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.ietf-mmusic-ice-sip-sdp] Petit-Huguenin, M. and A. Keranen, "Using Interactive Connectivity Establishment (ICE) with Session Description Protocol (SDP) offer/answer and Session Initiation Protocol (SIP)", draft-ietf-mmusic-ice-sip-sdp-04 (work in progress), October 2014.

[I-D.ietf-6man-ipv6-address-generation-privacy]
Cooper, A., Gont, F., and D. Thaler, "Privacy Considerations for IPv6 Address Generation Mechanisms", draft-ietf-6man-ipv6-address-generation-privacy-04 (work in progress), February 2015.

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 15 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.

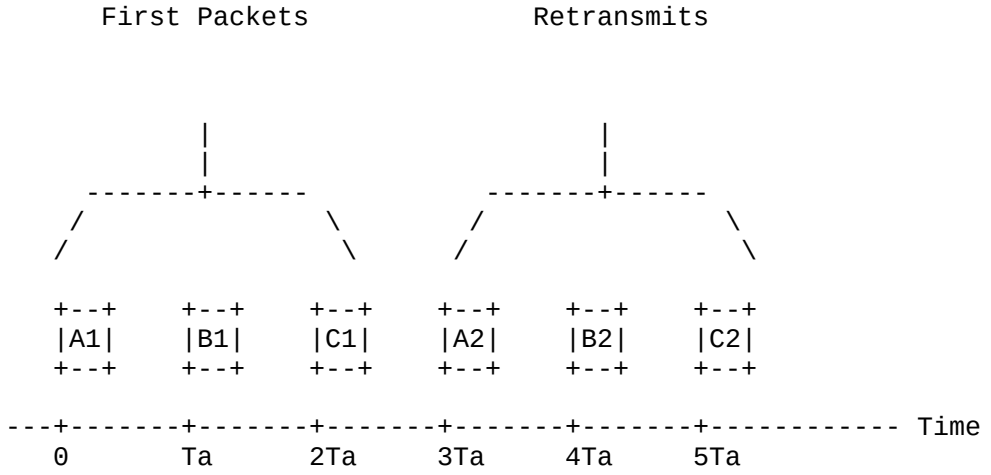
B.1. Pacing of STUN Transactions

STUN transactions used to gather candidates and to verify connectivity are paced out at an approximate rate of one new transaction every Ta milliseconds. Each transaction, in turn, has a retransmission timer RTO that is a function of Ta 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 Ta 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 [RFC5245] 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 Ta is designed so that, if a STUN packet were sent every Ta 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:

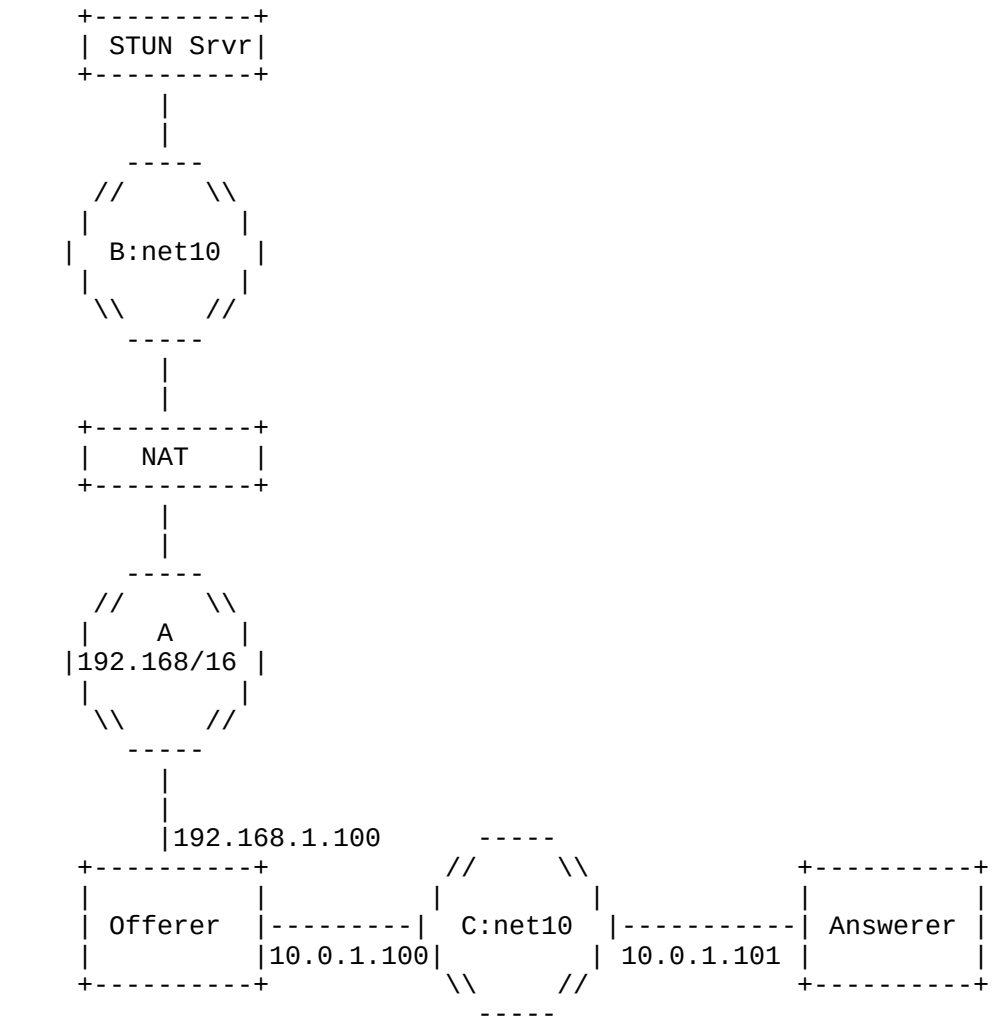


Figure 10: Identical Candidates with Different Bases

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:

```
pair priority = 2^32*MIN(G,D) + 2*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 15. 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 10. These keepalives make use of STUN when both endpoints are ICE capable. However, rather than using a Binding request transaction (which generates a response), the keepalives use an Indication. Why is that?

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

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

Authors' Addresses

Ari Keranen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: ari.keranen@ericsson.com

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

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

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

C. Holmberg
S. Loreto
G. Camarillo
Ericsson

March 5, 2015

Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)
draft-ietf-mmusic-sctp-sdp-14

Abstract

SCTP (Stream Control Transmission Protocol) is a transport protocol used to establish associations between two endpoints.

This specification describes how to describe SCTP associations using the Session Description Protocol (SDP), and defines the following new SDP Media Description protocol identifiers (proto values):'SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP'.

The specification also describes how to use the new proto values together with the SDP Offer/Answer mechanism in order to negotiate and establish SCTP associations, and how to indicate the SCTP application usage.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2015.

Table of Contents

1. Introduction

SDP (Session Description Protocol) [RFC4566] provides a general-purpose format for describing multimedia sessions in announcements or invitations. TCP-Based Media Transport in the Session Description Protocol (SDP) [RFC4145] specifies a general mechanism for describing and establishing TCP [RFC0793] streams. Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in SDP [RFC4572] extends RFC4145 [RFC4145] for describing TCP-based media streams that are protected using TLS.

SCTP (Stream Control Transmission Protocol) [RFC4960] is a transport protocol used to establish associations between two endpoints.

This specification defines how to describe SCTP associations using the Session Description Protocol (SDP) [RFC4566], and defines the following new SDP Media Description [RFC4566] protocol identifiers (proto values):'SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP'.

The specification also describes how to use the new proto values together with the SDP Offer/Answer mechanism [RFC3264] in order to negotiate and establish SCTP associations, and how to indicate the SCTP application usage.

NOTE: TLS is designed to run on top of a byte-stream oriented transport protocol providing a reliable, in-sequence delivery like TCP. [RFC6083] presents serious limitations with transporting SCTP on top of TLS. Therefore, defining a mechanism to negotiate media

streams transported using SCTP on top of TLS is outside the scope of this specification.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. SCTP Terminology

SCTP Association: A protocol relationship between SCTP endpoints, composed of the two SCTP endpoints and protocol state information including Verification Tags and the currently active set of Transmission Sequence Numbers (TSNs), etc. An association can be uniquely identified by the transport addresses used by the endpoints in the association.

SCTP Stream: A unidirectional logical channel established from one to another associated SCTP endpoint, within which all user messages are delivered in sequence except for those submitted to the unordered delivery service.

SCTP Transport address: A transport address is traditionally defined by a network-layer address, a transport-layer protocol, and a transport-layer port number. In the case of SCTP running over IP, a transport address is defined by the combination of an IP address and an SCTP port number (where SCTP is the transport protocol).

4. SDP Media Descriptions

4.1. General

This section defines the following new SDP Media Description (m-line) protocol identifiers (proto values) for describing an SCTP association: 'SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP'. The section also describes how an m- line, associated with the proto values, is created.

The following is the format for an 'm' line, as specified in RFC4566 [RFC4566]:

```
m=<media> <port> <proto> <fmt> ...
```

The 'SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP' proto values are similar to both the 'UDP' and 'TCP' proto values in that

they only describe the transport-layer protocol and not the upper-layer protocol.

NOTE: When the 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP' proto values are used, the underlying transport protocol is respectively UDP and TCP; SCTP is carried on top of DTLS which is on top of those transport-layer protocols.

The m- line fmt value, identifying the application-layer protocol, MUST be registered by IANA.

4.2. Protocol Identifiers

The new proto values are defined as below:

- The 'SCTP' proto value describes an SCTP association, as defined in [RFC4960].
- The 'SCTP/DTLS' proto value describes a Datagram Transport Layer Security (DTLS) [RFC6347] connection on top of an SCTP association, as defined in [RFC6083].
- The 'UDP/DTLS/SCTP' proto value describes an SCTP association on top of a DTLS connection on top of UDP, as defined in Section 7.
- The 'TCP/DTLS/SCTP' proto value describes an SCTP association on top of a DTLS connection on top of TCP, as defined in Section 8.

4.3. Media Format Management

[RFC4566] defines that specifications defining new proto values must define the rules by which their media format (fmt) namespace is managed. Use of an existing MIME subtype for the format is encouraged. If no MIME subtype exists, it is recommended that a suitable one is registered through the IETF process [RFC6838] [RFC4289] by production of, or reference to, a standards-track RFC that defines the transport protocol for the format.

An m- line with a proto value of 'SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP' always describe a single SCTP association.

In addition, such m- line MUST further indicate the application-layer protocol using an 'fmt' identifier. There MUST be exactly one 'fmt' value per m- line associated with the proto values defined in this specification. The "fmt" namespace associated with those proto values describes the generic application usage of the entire SCTP association, including the associated SCTP streams.

NOTE: A mechanism on how to describe, and manage, individual SCTP streams within an SCTP association, is outside the scope of this specification.

4.4. Syntax

4.4.1. General

This section defines the ABNF [RFC5234] for the SDP media description when associated with any of the proto values defined in this document.

This specification creates an IANA registry for 'association-usage' values.

4.4.2. ABNF

```
sctp-m-line = %x6d "="
 ("application" SP sctp-port SP "SCTP"      SP fmt CRLF) /
 ("application" SP sctp-port SP "SCTP/DTLS" SP fmt CRLF) /
 ("application" SP udp-port  SP "UDP/DTLS/SCTP" SP fmt CRLF) /
     ("application" SP tcp-port  SP "TCP/DTLS/SCTP" SP fmt CRLF)

sctp-port = port

udp-port = port

    tcp-port = port

fmt = association-usage

association-usage = token
```

4.5. Example

```
m=application 12345 UDP/DTLS/SCTP webrtc-datachannel
a=max-message-size: 100000
```

5. SDP 'sctp-port' Attribute

5.1. General

This section defines a new SDP media-level attribute, 'sctp-port'. The attribute can be associated with an SDP media description (m-line) with a 'UDP/DTLS/SCTP' or a 'TCP/DTLS/SCTP' proto value, in which case the m- line port value indicates the port of the

underlying transport-layer protocol (UDP or TCP), on which SCTP is carried, and the 'sctp-port' value indicates the SCTP port.

No default value is defined for the SDP sctp-port attribute. Therefore, if the attribute is not present, the associated m- line MUST be considered invalid.

Usage of the SDP sctp-port attribute with other proto values is not specified, and MUST be discarded if received.

5.2. Syntax

The ABNF for the SDP 'sctp-port' attribute is:

```
sctp-port-attr  =  "a=sctp-port:" port
    port            =  (1*5)DIGIT

    The SCTP port range is between 0 and 65535 (both included).
    Leading zeroes MUST NOT be used.
```

5.3. Mux Category

The mux category [I-D.ietf-mmusic-sdp-mux-attributes] for the SDP sctp-port' attribute is SPECIAL. Usage of the attribute is only applicable when associated with 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP' proto value m- lines.

As the usage of multiple SCTP associations on top of a single DTLS connection is outside the scope of this specification, no mux rules are specified for the 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP' proto values. Future extensions, that define how to negotiate multiplexing of multiple SCTP associations of top of a single DTLS connection, need to also define the mux rules for the attribute.

6. SDP 'max-message-size' Attribute

6.1. General

This section defines a new SDP media-level attribute, 'max-message-size'. The attribute can be associated with an m- line to indicate the maximum message size (indicated in bytes) that an SCTP endpoint is willing to receive on the SCTP association associated with the m-line. Different attribute values can be used in each direction.

The remote peer MUST assume that larger messages will be rejected by the SCTP endpoint. SCTP endpoints need to decide on appropriate

behavior in case a message that exceeds the maximum size needs to be sent.

If the SDP 'max-message-size' attribute contains a maximum message size value of zero, it indicates the SCTP endpoint will handle messages of any size, subject to memory capacity etc.

If the SDP 'max-message-size' attribute is not present, the default value is 64K.

NOTE: This specification only defines the usage of the SDP 'max-message-size' attribute when associated with an m- line containing one of the following proto values: 'SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP'. Usage of the attribute with other proto values needs to be defined in a separate specification.

6.2. Syntax

The ABNF for the SDP 'max-message-size' attribute is:

```
max-message-size-attr =  "a=max-message-size:" max-message-size
max-message-size      =  1*DIGIT
```

Leading zeroes MUST NOT be used.

6.3. Mux Category

The mux category for the SDP 'max-message-size' attribute is SPECIAL. The mux rules depends on the proto value of the associated m- line. If the proto value is 'SCTP' or 'SCTP/DTLS' the rules are identical to the rules associated with the TRANSPORT mux category.

As the usage of multiple SCTP associations on top of a single DTLS connection is outside the scope of this specification, no mux rules are specified for the 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP' proto values.

7. UDP/DTLS/SCTP Transport Realization

The UDP/DTLS/SCTP transport is realized as described below:

- SCTP on top of DTLS is realized according to the procedures defined in [I-D.ietf-tsvwg-sctp-dtls-encaps]; and
- DTLS on top of UDP is realized according to the procedures in defined in [RFC6347].

NOTE: While [I-D.ietf-tsvwg-sctp-dtls-encaps] allows multiple SCTP associations on top of a single DTLS connection, the procedures in this specification only supports the negotiation of a single SCTP association on top of any given DTLS connection.

8. TCP/DTLS/SCTP Transport Realization

The TCP/DTLS/SCTP transport is realized as described below:

- SCTP on top of DTLS is realized according to the procedures defined in [I-D.ietf-tsvwg-sctp-dtls-encaps]; and
- DTLS on top of TCP is realized using the framing method defined in [RFC4571], with DTLS packets being sent instead of RTP/RTCP packets, and SDP signaling according to the procedures defined in this specification.

NOTE: DTLS on top of TCP, without using the framing method defined in [RFC4571] is outside the scope of this specification. A separate proto value would need to be registered for such transport realization.

9. SCTP Association Management

9.1. General

The management of an SCTP association is identical to the management of a TCP connection. An SCTP endpoint MUST follow the rules in Section 6 of [RFC4145] to manage SCTP associations. Whether to use the SCTP ordered or unordered delivery service is up to the applications using the SCTP association, and this specification does not define a mechanism to indicate the type of delivery service using SDP.

9.2. SDP sendrecv/sendonly/recvonly/inactive Attribute

This specification does not define semantics for the SDP direction attributes [RFC4566]. Unless semantics of these attributes for an SCTP association usage have been defined, SDP direction attributes MUST be discarded if present.

9.3. SDP setup Attribute

9.3.1. General

The SDP setup attribute is used to determine the 'active/passive' status of the endpoints, following the procedures for TCP in [RFC4145].

9.3.2. SCTP Association Initiation

Both the 'active' and 'passive' endpoint MUST initiate the SCTP association, and MUST use the same SCTP port as client port and server port (in order to prevent two separate SCTP associations from being established).

NOTE: The procedure above is different from TCP, where only the 'active' endpoint initiates the TCP connection [RFC4145].

NOTE: If the underlying transport protocol is UDP or TCP (e.g. if the m- line proto value is 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP'), when the SCTP association is established it is assumed that any NAT traversal procedures for the underlying transport protocol has successfully been performed.

If the m- line proto value is 'TCP/DTLS/SCTP', the 'active' endpoint only MUST initiate the TCP connection, following the procedures in [RFC4145]. Both endpoints MUST still initiate the SCTP association on top of the TCP connection.

9.3.3. TLS Role Determination

If the m- line proto value is 'SCTP/DTLS', 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP', the 'active/passive' status is used to determine the (D)TLS roles of the endpoints. Following the procedures in [RFC4572], the 'active' endpoint will take the (D)TLS client role.

Once the DTLS connection has been established, the endpoints MUST NOT modify (as result of an offer/answer exchange) the TLS roles, or the 'active/passive' status, of the endpoints, unless the underlying transport protocol is also modified (e.g. if an IP address- or port value associated with the transport protocol is modified).

If the underlying transport protocol is modified, the endpoints MUST establish a new DTLS connection. In such case the 'active/passive' status of the endpoints will again be determined following the procedures in [RFC4145], and the new status will be used to determine the (D)TLS roles of the endpoints associated with the new DTLS connection.

NOTE: The procedure above is identical to the one defined for SRTP-DTLS in [RFC5763].

9.4. SDP connection Attribute

The SDP connection attribute is used following the procedures in [RFC4145], with the additional SCTP specific considerations described in this section.

If the m- line proto value is 'TCP/DTLS/SCTP', an SDP connection attribute associated with that m- line applies to both the SCTP association and the TCP connection. Therefore, an attribute 'new' value indicates that both a new SCTP association and new TCP connection have to be established, following the procedures in [RFC4145].

NOTE: This specification does not define a mechanism which allows re-establishing of a new SCTP association, while maintaining the underlying TCP connection.

The SDP connection attribute value does not automatically impact an existing DTLS connection. Section 9.3.3 describes in which cases a new DTLS connections will have to be re-established.

10. SDP Offer/Answer Procedures

10.1. General

This section defines the SDP Offer/Answer [RFC3264] procedures for negotiating and establishing an SCTP association. Unless explicitly stated, the procedures apply to all m- line proto values ('SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP') defined in this specification.

If the m- line proto value is 'SCTP/DTLS', 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP', each endpoint MUST provide a certificate fingerprint, using the SDP 'fingerprint' attribute [RFC4572], if the endpoint supports, and is willing to use, a cipher suite with an associated certificate.

The authentication certificates are interpreted and validated as defined in [RFC4572]. Self-signed certificates can be used securely, provided that the integrity of the SDP description is assured as defined in [RFC4572].

NOTE: The procedures apply to a specific m- line describing an SCTP association. If an offer or answer contains multiple m- lines describing SCTP associations, the procedures are applied separately to each m- line.

10.2. Generating the Initial SDP Offer

When the offerer creates an initial offer, the offerer:

- MUST, if the m- line proto value is 'SCTP/DTLS', 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP', associate an SDP setup attribute, with an 'actpass' value, with the m- line (see Section 9.3);
- MUST, if the m- line proto is 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP', associate an SDP 'sctp-port' attribute with the m- line (see Section 5);
- MUST associate an SDP 'connection' attribute, with a 'new' value, with the m- line (see Section 9.4); and
- MAY associate an SDP 'max-message-size' attribute with the m- line (see Section 6).

10.3. Generating the SDP Answer

When the answerer receives an offer, which contains an m- line describing an SCTP association, if the answerer accepts the m- line it:

- MUST insert a corresponding m- line in the answer, with an identical m- line proto value [RFC3264];
- MUST, if the m- line proto value is 'SCTP/DTLS', 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP', associate an SDP 'setup' attribute, with an 'active' or 'passive' value, with the m- line (see Section 9.3);
- MUST, if the m- line proto is 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP', associate an SDP 'sctp-port' attribute with the m- line (see Section 5); and
- MAY associate an SDP 'max-message-size' attribute with the m- line (see Section 6). The attribute value in the answer is independent from the value (if present) in the corresponding m- line of the offer.

Once the answerer has sent the answer, the answerer:

- MUST, if an SCTP association associated with the m- line has yet not been established, or if an existing SCTP association is to be re-established, initiate the establishing of the SCTP association; and

- MUST, if the answerer is the 'active' endpoint, and if an DTLS connection associated with the m- line is to be established (or re-established), initiate the establishing of the DTLS connection (by sending a ClientHello message).

If the answerer does not accept the m- line in the offer, it MUST assign a zero port value to the corresponding m- line in the answer. In addition, the answerer MUST NOT establish an SCTP association, or a DTLS connection, associated with the m- line.

10.4. Offerer Processing of the SDP Answer

When the offerer receives an answer, which contains an m- line with a non-zero port value, describing an SCTP association, the offerer:

- MUST, if the offerer is the 'active' endpoint, if the m- line proto value is 'TCP/DTLS/SCTP', and if a TCP connection used to carry the SCTP association has not yet been established (or if an existing TCP connection is to be re-established), initiate the establishing of the TCP connection;

- MUST, if an SCTP association associated with the m- line has not yet been established (or if an existing SCTP association is to be re-established), initiate the establishing of the SCTP association; and

- MUST, if the offerer is the 'active' endpoint, and if a DTLS connection associated with the m- line is to be established (or if an existing DTLS connection is to be re-established), initiate the establishing of the DTLS connection (by sending a ClientHello message).

- NOTE: If the m- line proto value is 'UDP/DTLS/SCTP' or 'TCP/DTLS/SCTP', the underlying DTLS connection needs to be established before the SCTP association can be established.

If the m- line in the answer contains a zero port value, the offerer MUST NOT establish a TCP connection, an SCTP association, or a DTLS connection, associated with the m- line.

10.5. Modifying the Session

When an offerer sends an updated offer, in order to modify a previously established SCTP association, it follows the procedures in Section 10.2, with the following exceptions:

- Unless the offerer wants to re-establish an existing SCTP association, the offerer MUST associate an SDP connection attribute, with an 'existing' value, with the m- line; and

- If the offerer wants to disable a previously established SCTP association, it MUST assign a zero port value to the m- line associated with the SCTP association, following the procedures in [RFC3264].

11. Multihoming Considerations

SCTP supports multihoming. An SCTP endpoint is considered multihomed if it has more than one IP address on which SCTP can be used. An SCTP endpoint inform the remote peer about its IP addresses using the address parameters in the INIT/INIT-ACK chunk. Therefore, when SDP is used to describe an SCTP association, while the "c=" line contains the address which was used to negotiate the SCTP association, multihomed SCTP endpoints might end up using other IP addresses.

If an endpoint removes the IP address [RFC5061] that it offered in the SDP "c=" line associated with the SCTP association, it MUST send a new Offer, in which the "c=" line contains an IP address which is valid within the SCTP association.

NOTE: In some network environments, intermediaries performing gate- and firewall control using the address information in the SDP "c=" and "m=" lines to authorize media, and will not pass media sent using other addresses. In such network environments, if an SCTP endpoints wants to change the address information on which media is sent and received, it needs to send an updated Offer, in which the SDP "c=" and "m=" lines contain the new address information.

Multihoming is not supported when sending SCTP on top of DTLS, as DTLS does not expose address management of the underlying transport protocols (UDP or TCP) to its upper layer.

12. NAT Considerations

12.1. General

SCTP features not present in UDP or TCP, including the checksum (CRC32c) value calculated on the whole packet (rather than just the header), and multihoming, introduce new challenges for NAT traversal. [I-D.ietf-behave-sctpnat] defines an SCTP specific variant of NAT, which provides similar features of Network Address and Port Translation (NAPT).

Current NATs typically do not support SCTP. [RFC6951] defines a mechanism for sending SCTP on top of UDP, which makes it possible to use SCTP with NATs and firewalls that do not support SCTP.

12.2. ICE Considerations

At the time of writing this specification, no procedures have been defined for using ICE (Interactive Connectivity Establishment) [RFC5245] together with SCTP as transport layer protocol. Such procedures, including the associated SDP Offer/Answer procedures, are outside the scope of this specification, and might be defined in a future specification.

When the transport layer protocol is UDP (in case of an SCTP association on top of a DTLS connection on top of UDP), if ICE is used, the ICE procedures defined in [RFC5245] are used.

When the transport layer protocol is TCP (in case of an SCTP association on top of a DTLS connection on top of TCP), if ICE is used, the ICE procedures defined in [RFC6544] are used.

13. Examples

13.1. Establishment of UDP/DTLS/SCTP association

SDP Offer:

```
        m=application 54111 UDP/DTLS/SCTP webrtc-datachannel
        c=IN IP4 192.0.2.1
    a=setup:actpass
    a=connection:new
        a=sctp-port:5000
        a=max-message-size:100000

    - The offerer indicates that the usage of the
          UDP/DTLS/SCTP association will be as defined
          for the 'webrtc-datachannel' format value.
        - The offerer UDP port value is 54111.
        - The offerer SCTP port value is 5000.
        - The offerer indicates that it can take either the
          active or the passive role.
```

SDP Answer:

```
        m=application 64300 UDP/DTLS/SCTP webrtc-datachannel
        c=IN IP4 192.0.2.2
    a=setup:passive
        a=sctp-port:6000
        a=max-message-size:100000

    - The answerer UDP port value is 64300.
        - The answerer SCTP port value is 6000.
        - The answerer takes the passive role.
```

14. Security Considerations

[RFC4566] defines general SDP security considerations, while [RFC3264], [RFC4145] and [RFC4572] define security considerations when using the SDP offer/answer mechanism to negotiate media streams.

[RFC4960] defines general SCTP security considerations, while [RFC6083] defines security considerations when using DTLS on top of SCTP.

This specification does not introduce new security considerations in addition to those defined in the specifications listed above.

15. IANA Considerations

15.1. New SDP proto values

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

This document updates the "Session Description Protocol (SDP) Parameters" registry, following the procedures in [RFC4566], by adding the following values to the table in the SDP "proto" field registry:

Type	SDP Name	Reference
proto	SCTP	[RFCXXXX]
proto	SCTP/DTLS	[RFCXXXX]
proto	UDP/DTLS/SCTP	[RFCXXXX]
proto	TCP/DTLS/SCTP	[RFCXXXX]

Table 1: SDP "proto" field values

15.2. New SDP Attributes

15.2.1. sctp-port

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

This document defines a new SDP media-level attribute,'sctp-port', as follows:

```
Attribute name:     sctp-port
Type of attribute:  media
Mux category:       SPECIAL
Subject to charset: No
Purpose:            Indicate the SCTP port value associated
                 with the SDP Media Description.
Appropriate values: Integer
Contact name:       Christer Holmberg
Contact e-mail:     christer.holmberg@ericsson.com
Reference:          RFCXXXX
```

15.2.2. max-message-size

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

This document defines a new SDP media-level attribute,'max-message-size', as follows:

```
Attribute name:     max-message-size
Type of attribute:  media
Mux category:       SPECIAL
Subject to charset: No
Purpose:            Indicate the maximum message size that
                an SCTP endpoint is willing to receive
                on the SCTP association associated
                with the SDP Media Description.
Appropriate values: Integer
Contact name:       Christer Holmberg
Contact e-mail:     christer.holmberg@ericsson.com
Reference:          RFCXXXX
```

15.3. association-usage Name Registry

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

This specification creates a new IANA registry, following the procedures in [RFC5226], for the "fmt" namespace associated with the 'SCTP', 'SCTP/DTLS', 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP' protocol identifiers. Each "fmt" value describes the usage of an entire SCTP association, including all SCTP streams associated with the SCTP association.

NOTE: Usage indication of individual SCTP streams is outside the scope of this specification.

The "fmt" value, "association-usage", used with these "proto" is required. It is defined in [Section 4].

As part of this registry, IANA maintains the following information:

association-usage name: The identifier of the subprotocol, as will be used as the "fmt" value.

association-usage reference: A reference to the document in which the association-usage is defined.

association-usage names are to be subject to the "First Come First Served" IANA registration policy [RFC5226].

IANA is asked to add initial values to the registry.

name	Reference
webrtc-datachannel	draft-ietf-rtcweb-data-protocol-xx

[RFC EDITOR NOTE: Please hold the publication of this draft until draft-ietf-rtcweb-data-protocol has been published as an RFC. Then, replace the reference to draft-ietf-rtcweb-data-protocol with the RFC number.]

Figure 1

16. Acknowledgments

The authors wish to thank Harald Alvestrand, Randell Jesup, Paul Kyzivat, Michael Tuexen, Juergen Stoetzer-Bradler, Flemming Andreasen and Ari Keranen for their comments and useful feedback.

17. Change Log

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

Changes from draft-ietf-mmusic-sctp-sdp-13

- Changes based on comments from Paul Kyzivat.
- - Text preventing usage of well-known ports removed.
- - Editorial clarification.

Changes from draft-ietf-mmusic-sctp-sdp-12

- Mux category rules added for new SDP attributes.
- Reference to draft-ietf-mmusic-sdp-mux-attributes added.
- Changes based on comments from Roman Shpount:
- - Specify that fingerprint or setup roles must not be modified, unless underlying transport protocol is also modified.

- Changes based on comments from Ari Keranen:
- - Editorial corrections.
- Changes based on comments from Flemming Andreasen:
- - Clarify that, if UDP/DTLS/SCTP or TCP/DTLS/SCTP is used, the DTLS connection is established before the SCTP association.
- - Clarify that max-message-size value is given in bytes, and that different values can be used per direction.
- - Section on fmtp attribute removed.
- - Editorial corrections.

Changes from draft-ietf-mmusic-sctp-sdp-11

- Example added.

Changes from draft-ietf-mmusic-sctp-sdp-10

- SDP max-message-size attribute added to IANA considerations.
- Changes based on comments from Paul Kyzivat:
- - Text about max message size removed from fmtp attribute section.

Changes from draft-ietf-mmusic-sctp-sdp-09

- 'DTLS/SCTP' split into 'UDP/DTLS/SCTP' and 'TCP/DTLS/SCTP'
- Procedures for realizing UDP/DTLS/SCTP- and TCP/DTLS/SCTP transports added.

Changes from draft-ietf-mmusic-sctp-sdp-08

- Default SCTP port removed:
- - Usage of SDP sctp-port attribute mandatory.
- SDP max-message-size attribute defined:
- - Attribute definition.
- - SDP Offer/Answer procedures.
- Text about SDP direction attributes added.

- Text about TLS role determination added.

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.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

[RFC4289] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, December 2005.

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

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, 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.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[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.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

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

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[I-D.ietf-tsvwg-sctp-dtls-encaps] Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "DTLS Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-dtls-encaps-09 (work in progress), January 2015.

[I-D.ietf-mmusic-sdp-mux-attributes] Nandakumar, S., "A Framework for SDP Attributes when Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-08 (work in progress), January 2015.

18.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

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

[RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011.

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

[I-D.ietf-behave-sctpnat]
Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation", draft-ietf-behave-sctpnat-09 (work in progress), September 2013.

Authors' Addresses

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Salvatore.Loreto@ericsson.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

MMUSIC Working Group
Internet-Draft
Updates: 3264 (if approved)
Intended status: Standards Track
Expires: September 10, 2015

C. Holmberg
Ericsson
H. Alvestrand
Google
C. Jennings
Cisco

March 9, 2015

Negotiating Media Multiplexing Using the Session Description Protocol (SDP)

draft-ietf-mmusic-sdp-bundle-negotiation-18.txt

Abstract

This specification defines a new Session Description Protocol (SDP) Grouping Framework extension, 'BUNDLE'. The extension can be used with the SDP Offer/Answer mechanism to negotiate the usage of a single address:port combination (BUNDLE address) for receiving media, referred to as bundled media, associated with multiple SDP media descriptions ("m=" lines).

To assist endpoints in negotiating the use of bundle this specification defines a new SDP attribute, 'bundle-only', which can be used to request that specific media is only used if bundled. This specification also updates sections 5.1, 8.1 and 8.2 of RFC 3264 to allow an answerer to assign a non-zero port value to an "m=" line in an SDP answer, even if the "m=" line in the associated SDP offer contained a zero port value.

There are multiple ways to correlate the bundled RTP packets with the appropriate media descriptions. This specification defines a new RTCP source description (SDES) item and a new RTP header extension that provides an additional way to do this correlation by using them to carry a value that associates the RTP/RTCP packets with a specific media description.

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/.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a way to use a single address:port combination (BUNDLE address) for receiving media associated with multiple SDP media descriptions ("m=" lines).

This specification defines a new SDP Grouping Framework [RFC5888] extension called 'BUNDLE'. The extension can be used with the Session Description Protocol (SDP) Offer/Answer mechanism [RFC3264] to negotiate the usage of a BUNDLE group. Within the BUNDLE group, a BUNDLE address is used for receiving media associated with multiple "m=" lines. This is referred to as bundled media.

The offerer and answerer [RFC3264] use the BUNDLE extension to negotiate the BUNDLE addresses, one for the offerer (offerer BUNDLE address) and one for the answerer (answerer BUNDLE address), to be used for receiving the bundled media associated with a BUNDLE group. Once the offerer and the answerer have negotiated a BUNDLE group, they assign their respective BUNDLE address to each "m=" line in the BUNDLE group. The BUNDLE addresses are used to receive all media associated with the BUNDLE group.

The use of a BUNDLE group and a BUNDLE address also allows the usage of a single set of Interactive Connectivity Establishment (ICE) [RFC5245] candidates for multiple "m=" lines.

This specification also defines a new SDP attribute, 'bundle-only', which can be used to request that specific media is only used if kept within a BUNDLE group.

As defined in RFC 4566 [RFC4566], the semantics of assigning the same port value to multiple "m=" lines 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 an extension.

This specification also updates sections 5.1, 8.1 and 8.2 of RFC 3264 [RFC3264]. The update allows an answerer to assign a non-zero port value to an "m=" line in an SDP answer, even if the "m=" line in the associated SDP offer contained a zero port value.

This specification also defines a new Real-time Transport Protocol (RTP) [RFC3550] SDES item and a new RTP header extension that can be used to carry a value that associates RTP/RTCP packets with a specific media description. This can be used to correlate a RTP packet with the correct media.

SDP bodies can contain multiple BUNDLE groups. A given BUNDLE address MUST only be associated with a single BUNDLE group. The procedures in this specification apply independently to a given BUNDLE group. All RTP based media flows associated with a single BUNDLE group belong to a single RTP session [RFC3550].

The BUNDLE extension is backward compatible. Endpoints that do not support the extension are expected to generate offers and answers without an SDP 'group:BUNDLE' attribute, and are expected to assign a unique address to each "m=" line within an offer and answer, according to the procedures in [RFC4566] and [RFC3264]

2. Terminology

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

Unique address: An IP address and port combination that is assigned to only one "m=" line in an offer or answer.

Shared address: An IP address and port combination that is assigned to multiple "m=" lines within an offer or answer.

Offerer BUNDLE-tag: The first identification-tag in a given SDP 'group:BUNDLE' attribute identification-tag list in an offer.

Answerer BUNDLE-tag: The first identification-tag in a given SDP 'group:BUNDLE' attribute identification-tag list in an answer.

Offerer BUNDLE address: Within a given BUNDLE group, an IP address and port combination used by an offerer to receive all media associated with each "m=" line within the BUNDLE group.

Answerer BUNDLE address: Within a given BUNDLE group, an IP address and port combination used by an answerer to receive all media associated with each "m=" line within the BUNDLE group.

BUNDLE group: A set of "m=" lines, created using an SDP Offer/Answer exchange, which uses the same BUNDLE address for receiving media.

Bundled "m=" line: An "m=" line, whose identification-tag is placed in an SDP 'group:BUNDLE' attribute identification-tag list in an offer or answer.

Bundle-only "m=" line: A bundled "m=" line with an associated SDP 'bundle-only' attribute.

Bundled media: All media associated with a given BUNDLE group.

Initial offer: The first offer, within an SDP session (e.g. a SIP dialog when the Session Initiation Protocol (SIP) [RFC3261] is used to carry SDP), in which the offerer indicates that it wants to create a given BUNDLE group.

Subsequent offer: An offer which contains a BUNDLE group that has been created as part of a previous offer/answer exchange.

Identification-tag: A unique token value that is used to identify an "m=" line. The SDP 'mid' attribute [RFC5888], associated with an "m=" line, carries an unique identification-tag. The session-level SDP 'group' attribute [RFC5888] carries a list of identification-tags, identifying the "m=" lines associated with that particular 'group' attribute.

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]. Declarative usage of SDP is out of scope of this document, and is thus undefined.

5. SDP Grouping Framework BUNDLE Extension

This section defines a new SDP Grouping Framework extension [RFC5888], 'BUNDLE'. The BUNDLE extension can be used with the SDP Offer/Answer mechanism to negotiate the usage of a single address:port combination (BUNDLE address) for receiving bundled media.

A single address:port combination is also used for sending bundled media. The address:port combination used for sending bundled media MAY be the same as the BUNDLE address, used to receive bundled media, depending on whether symmetric RTP is used. A given address:port combination MUST NOT be used for sending media associated with multiple BUNDLE groups.

All media associated with a BUNDLE group share a single 5-tuple, i.e. in addition to using a single address:port combination all bundled media MUST be transported using the same transport-layer protocol (e.g. UDP or TCP).

The BUNDLE extension is indicated using an SDP 'group' attribute with a "BUNDLE" semantics value [RFC5888]. An identification-tag is assigned to each bundled "m=" line, and each identification-tag is listed in the SDP 'group:BUNDLE' attribute identification-tag list. Each "m=" line, whose identification-tag is listed in the identification-tag list, is associated with a given BUNDLE group.

SDP bodies can contain multiple BUNDLE groups. Any given bundled "m=" line MUST NOT be associated with more than one BUNDLE group.

Section 8 defines the detailed SDP Offer/Answer procedures for the BUNDLE extension.

6. SDP 'bundle-only' Attribute

This section defines a new SDP media-level attribute [RFC4566], 'bundle-only'.

Name: bundle-only

Value:

Usage Level: media

Charset Dependent: no

Example:

a=bundle-only

In order to ensure that an answerer that does not supports the BUNDLE extension always rejects a bundled "m=" line, the offerer can assign a zero port value to the "m=" line. According to [RFC4566] an answerer will reject such "m=" line. By associating an SDP 'bundle-only' attribute with such "m=" line, the offerer can request that the answerer accepts the "m=" line if the answerer supports the Bundle extension, and if the answerer keeps the "m=" line within the associated BUNDLE group.

NOTE: Once an offerer BUNDLE address has been selected, the offerer can ensure that an bundled "m=" line is accepted by the answerer only if the answerer keeps the "m=" line within the associated BUNDLE group by assigning the offerer BUNDLE address to the "m=" line. If the answerer does not keep that "m=" line within the BUNDLE group, the answerer will reject it. Therefore, the SDP 'bundle-only' attribute is not needed in such cases

The usage of the 'bundle-only' attribute is only defined for a bundled "m=" line with a zero port value, within an offer. Other usage is unspecified.

Section 8 defines the detailed SDP Offer/Answer procedures for the 'bundle-only' attribute.

7. SDP Information Considerations

7.1. General

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

7.2. Connection Data (c=)

The "c=" line nettype value [RFC4566] associated with a bundled "m=" line MUST be 'IN'.

The "c=" line addrtype value [RFC4566] associated with a bundled "m=" line MUST be 'IP4' or 'IP6'. The same value MUST be associated with each "m=" line.

NOTE: Extensions to this specification can specify usage of the BUNDLE mechanism for other nettype and addrtype values than the ones listed above.

7.3. Bandwidth (b=)

The proposed bandwidth for a bundled "m=" line SHOULD be calculated in the same way as for a non-bundled "m=" line.

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

The total proposed bandwidth for an offer or answer is the sum of the proposed bandwidth for each "m=" line (bundled and non-bundled) within the offer or answer.

7.4. Attributes (a=)

An offerer and answerer MUST use the rules and restrictions defined in [I-D.mmusic-sdp-mux-attributes] for when associating SDP attributes with bundled "m=" lines.

8. SDP Offer/Answer Procedures

8.1. General

This section describes the SDP Offer/Answer [RFC3264] procedures for:

- Negotiating and creating of a BUNDLE group;
- Selecting the BUNDLE addresses (offerer BUNDLE address and answerer BUNDLE address);
- Adding an "m=" line to a BUNDLE group;
- Moving an "m=" line out of a BUNDLE group; and
- Disabling an "m=" line within a BUNDLE group.

The generic rules and procedures defined in [RFC3264] and [RFC5888] also apply to the BUNDLE extension. For example, if an offer is rejected by the answerer, the previously negotiated SDP parameters and characteristics (including those associated with a BUNDLE group) apply. Hence, if an offerer generates an offer in which the offerer wants to create a BUNDLE group, and the answerer rejects the offer, the BUNDLE group is not created.

The procedures in this section are independent of the media type or "m=" line proto value represented by a bundled "m=" line. Section 10 defines additional considerations for RTP based media. Section 6 defines additional considerations for the usage of the SDP 'bundle-only' attribute. Section 11 defines additional considerations for the usage of Interactive Connectivity Establishment (ICE) [RFC5245] mechanism .

The offerer and answerer MUST follow the rules and restrictions defined in Section 7 when creating offers and answers.

SDP offers and answers can contain multiple BUNDLE groups. The procedures in this section apply independently to a given BUNDLE group.

8.2. Generating the Initial SDP Offer

8.2.1. General

When an offerer generates an initial offer, in order to create a BUNDLE group, it MUST:

- Assign a unique address to each "m=" line within the offer, following the procedures in [RFC3264], unless the media line is a 'bundle-only' "m=" line (see below);
- Add an SDP 'group:BUNDLE' attribute to the offer;
- Place the identification-tag of each bundled "m=" line in the SDP 'group:BUNDLE' attribute identification-tag list; and
- Indicate which unique address the offerer suggests as the offerer BUNDLE address [Section 8.2.2].

If the offerer wants to request that the answerer accepts a given bundled "m=" line only if the answerer keeps the "m=" line within the BUNDLE group, the offerer MUST:

- Associate an SDP 'bundle-only' attribute [Section 8.2.2] with the "m=" line; and

- Assign a zero port value to the "m=" line.

NOTE: If the offerer assigns a zero port value to an "m=" line, but does not also associate an SDP 'bundle-only' attribute with the "m=" line, it is an indication that the offerer wants to disable the "m=" line [Section 8.5.5].

[Section 16.1] shows an example of an initial offer.

8.2.2. Suggesting the offerer BUNDLE address

In the offer, the address assigned to the "m=" line associated with the offerer BUNDLE-tag indicates the address that the offerer suggests as the offerer BUNDLE address.

8.3. Generating the SDP Answer

8.3.1. General

When an answerer generates an answer, which contains a BUNDLE group, the following general SDP grouping framework restrictions, defined in [RFC5888], also apply to the BUNDLE group:

- The answerer MUST NOT include a BUNDLE group in the answer, unless the offerer requested the BUNDLE group to be created in the associated offer; and

- The answerer MUST NOT include an "m=" line within a BUNDLE group, unless the offerer requested the "m=" line to be within that BUNDLE group in the associated offer.

If the answer contains a BUNDLE group, the answerer MUST:

- Select an Offerer BUNDLE Address [Section 8.3.2]; and

- Select an Answerer BUNDLE Address [Section 8.3.3];

The answerer is allowed to select a new Answerer BUNDLE address each time it generates an answer to an offer.

If the answerer does not want to keep an "m=" line within a BUNDLE group, it MUST:

- Move the "m=" line out of the BUNDLE group [Section 8.3.4]; or

- Reject the "m=" line [Section 8.3.5];

If the answerer keeps a bundle-only "m=" line within the BUNDLE group, it follows the procedures (assigns the answerer BUNDLE address to the "m=" line etc) for any other "m=" line kept within the BUNDLE group.

If the answerer does not want to keep a bundle-only "m=" line within the BUNDLE group, it MUST reject the "m=" line [Section 8.3.5].

The answerer MUST NOT associate an SDP 'bundle-only' attribute with any "m=" line in an answer.

NOTE: If a bundled "m=" line in an offer contains a zero port value, but the "m=" line does not contain an SDP 'bundle-only' attribute, it is an indication that the offerer wants to disable the "m=" line [Section 8.5.5].

8.3.2. Answerer Selection of Offerer Bundle Address

In an offer, the address (unique or shared) assigned to the bundled "m=" line associated with the offerer BUNDLE-tag indicates the address that the offerer suggests as the offerer BUNDLE address [Section 8.2.2]. The answerer MUST check whether that "m=" line fulfills the following criteria:

- The answerer will not move the "m=" line out of the BUNDLE group [Section 8.3.4];
- The answerer will not reject the "m=" line [Section 8.3.5]; and
- The "m=" line does not contain a zero port value.

If all of the criteria above are fulfilled, the answerer MUST select the address associated with the "m=" line as the offerer BUNDLE address. In the answer, the answerer BUNDLE-tag represents the "m=" line, and the address associated with the "m=" line in the offer becomes the offerer BUNDLE address.

If one or more of the criteria are not fulfilled, the answerer MUST select the next identification-tag in the identification-tag list, and perform the same criteria check for the "m=" line associated with that identification-tag. If there are no more identification-tags in the identification-tag list, the answerer MUST NOT create the BUNDLE group. In addition, unless the answerer rejects the whole offer, the answerer MUST apply the answerer procedures for moving an "m=" line out of a BUNDLE group [Section 8.3.4] to each bundled "m=" line in the offer when creating the answer.

[Section 16.1] shows an example of an offerer BUNDLE address selection.

8.3.3. Answerer Selection of Answerer BUNDLE Address

When the answerer selects a BUNDLE address for itself, referred to as the answerer BUNDLE address, it MUST assign that address to each bundled "m=" line within the created BUNDLE group in the answer.

The answerer MUST NOT assign the answerer BUNDLE address to an "m=" line that is not within the BUNDLE group, or to an "m=" line that is within another BUNDLE group.

[Section 16.1] shows an example of an answerer BUNDLE address selection.

8.3.4. Moving A Media Description Out Of A BUNDLE Group

When an answerer moves a "m=" line out of a BUNDLE group, it assigns an address to the "m=" line in the answer based on the following rules:

- In the associated offer, if the "m=" line contains a shared address (e.g. a previously selected offerer BUNDLE address), the answerer MUST reject the moved "m=" line [Section 8.3.5];

- In the associated offer, if the "m=" line contains a unique address, the answerer MUST assign a unique address also to the "m=" line in the answer; or

- In the associated offer, if an SDP ’bundle-only’ attribute is associated with the "m=" line, and if the "m=" line contains a zero port value, the answerer MUST reject the "m=" line [Section 8.3.5].

In addition, in either case above, the answerer MUST NOT place the identification-tag, associated with the moved "m=" line, in the SDP ’group’ attribute identification-tag list associated with the BUNDLE group.

8.3.5. Rejecting A Media Description In A BUNDLE Group

When an answerer rejects an "m=" line, it MUST assign an address with a zero port value to the "m=" line in the answer, according to the procedures in [RFC4566].

In addition, the answerer MUST NOT place the identification-tag, associated with the rejected "m=" line, in the SDP 'group' attribute identification-tag list associated with the BUNDLE group.

8.4. Offerer Processing of the SDP Answer

8.4.1. General

When an offerer receives an answer, if the answer contains a BUNDLE group, the offerer MUST check that any bundled "m=" line in the answer was indicated as bundled in the associated offer. If there is no mismatch, the offerer MUST use the offerer BUNDLE address, selected by the answerer [Section 8.3.2], as the address for each bundled "m=" line.

NOTE: As the answerer might reject one or more bundled "m=" lines, or move a bundled "m=" line out of a BUNDLE group, each bundled "m=" line in the offer might not be indicated as bundled in the answer.

If the answer does not contain a BUNDLE group, the offerer MUST process the answer as a normal answer.

8.4.2. Bundle Address Synchronization (BAS)

When an offerer receives an answer, if the answer contains a BUNDLE group, the offerer MUST check whether the offerer BUNDLE address, selected by the answerer [Section 8.3.2], matches what was assigned to each bundled "m=" line (excluding any bundled "m=" line that was rejected, or moved out of the BUNDLE group, by the answerer) in the associated offer. If there is a mismatch, the offerer SHOULD as soon as possible generate a subsequent offer, in which it assigns the offerer BUNDLE address to each bundled "m=" line. Such offer is referred to as a Bundle Address Synchronization (BAS) offer.

A BAS offer is typically sent in the following scenarios:

- The offerer receives an answer to an initial offer, as the bundled "m=" lines in the initial offer always contain unique addresses [Section 8.2]; or

- The offerer receives an answer to an offer, in which a new bundled "m=" line has been added to the BUNDLE group [Section 8.5.3], and the offerer assigned a unique address to the bundled "m=" line in the offer.

The offerer is allowed to modify any SDP parameter in the BAS offer.

NOTE: It is important that the BAS offer gets accepted by the answerer. For that reason the offerer needs to consider the necessity to modify SDP parameters in the BAS offer, in such a way that could trigger the answerer to reject the BAS offer. Disabling "m=" lines, or reducing the number of codecs, in a BAS offer is considered to have a low risk of being rejected.

NOTE: The main purpose of the BAS offer is to ensure that intermediaries, that might not support the BUNDLE extension, have correct information regarding the address that is going to be used to transport the bundled media.

[Section 16.1] shows an example of a BAS offer.

8.5. Modifying the Session

8.5.1. General

When an offerer generates a subsequent offer, it MUST assign the previously selected offerer BUNDLE address [Section 8.3.2], to each bundled "m=" line (including any bundle-only "m=" line), except if:

- The offerer suggests a new offerer BUNDLE address [Section 8.5.2];
- The offerer wants to add a bundled "m=" line to the BUNDLE group [Section 8.5.3];
- The offerer wants to move a bundled "m=" line out of the BUNDLE group [Section 8.5.4]; or
- The offerer wants to disable the bundled "m=" line [Section 8.5.5].

In addition, the offerer MUST select an offerer BUNDLE-tag [Section 8.2.2] associated with the previously selected offerer BUNDLE address, unless the offerer suggests a new offerer BUNDLE address.

8.5.2. Suggesting a new offerer BUNDLE address

When an offerer generates an offer, in which it suggests a new offerer BUNDLE address [Section 8.2.2], the offerer MUST:

- Assign the address (shared address) to each "m=" line within the BUNDLE group; or
- Assign the address (unique address) to one bundled "m=" line.

NOTE: If the offerer assigns a unique address, there might be a need to send a subsequent BAS offer [Section 8.4.2] once the offerer has received the associated answer.

In addition, the offerer MUST indicate that the address is the new suggested offerer BUNDLE address [Section 8.2.2].

NOTE: Unless the offerer assigns the new suggested offerer BUNDLE address to each bundled "m=" line, it can assign unique addresses to any number of bundled "m=" lines (and the previously selected offerer BUNDLE address to any remaining bundled "m=" line) if it wants to suggest multiple alternatives for the new offerer BUNDLE address.

8.5.3. Adding a media description to a BUNDLE group

When an offerer generates an offer, in which it wants to add a bundled "m=" line to a BUNDLE group, the offerer MUST:

- Assign a unique address to the "m=" line;
- Assign the previously selected offerer BUNDLE address to the "m=" line; or
- If the offerer assigns a new (shared address) suggested offerer BUNDLE address to each bundled "m=" line [Section 8.5.2], also assign that address to the added "m=" line.

In addition, the offerer MUST extend the SDP 'group:BUNDLE' attribute identification-tag list with the BUNDLE group [Section 8.2.2] by adding the identification-tag associated with the added "m=" line to the list.

NOTE: Assigning a unique address to the "m=" line allows the answerer to move the "m=" line out of the BUNDLE group [Section 8.3.4], without having to reject the "m=" line.

If the offerer assigns a unique address to the added "m=" line, and if the offerer suggests that address as the new offerer BUNDLE address [Section 8.5.2], the offerer BUNDLE-tag MUST represent the added "m=" line [Section 8.2.2].

If the offerer assigns a new suggested offerer BUNDLE address to each bundled "m=" line [Section 8.5.2], including the added "m=" line, the offerer BUNDLE-tag MAY represent the added "m=" line [Section 8.2.2].

[Section 16.3] shows an example where an offerer sends an offer in order to add a bundled "m=" line to a BUNDLE group.

8.5.4. Moving A Media Description Out Of A BUNDLE Group

When an offerer generates an offer, in which it wants to move a bundled "m=" line out of a BUNDLE group it was added to in a previous offer/answer transaction, the offerer:

- MUST assign a unique address to the "m=" line; and
- MUST NOT place the identification-tag associated with the "m=" line in the SDP 'group:BUNDLE' attribute identification-tag list associated with the BUNDLE group.

NOTE: If an "m=" line, when being moved out of a BUNDLE group, is added to another BUNDLE group, the offerer applies the procedures in [Section 8.5.3] to the "m=" line.

[Section 16.4] shows an example of an offer for moving an "m=" line out of a BUNDLE group.

8.5.5. Disabling A Media Description In A BUNDLE Group

When an offerer generates an offer, in which it wants to disable a bundled "m=" line (added to the BUNDLE group in a previous offer/answer transaction), the offerer:

- MUST assign an address with a zero port value to the "m=" line, following the procedures in [RFC4566]; and
- MUST NOT place the identification-tag associated with the "m=" line in the SDP 'group:BUNDLE' attribute identification-tag list associated with the BUNDLE group.

[Section 16.5] shows an example of an offer for disabling an "m=" line within a BUNDLE group.

9. Protocol Identification

9.1. General

Each "m=" line within a BUNDLE group MUST use the same transport-layer protocol. If bundled "m=" lines use different protocols on top of the transport-layer protocol, there MUST exist a publicly available specification which describes a mechanism, for this particular protocol combination, how to associate a received packet with the correct protocol.

In addition, if a received packet can be associated with more than one bundled "m=" line, there MUST exist a publically available

specification which describes a mechanism for associating the received packet with the correct "m=" line.

9.2. STUN, DTLS, SRTP

Section 5.1.2 of [RFC5764] describes a mechanism to identify the protocol of a received packet among the STUN, DTLS and SRTP protocols (in any combination). If an offer or answer includes bundled "m=" lines that represent these protocols, the offerer or answerer MUST support the mechanism described in [RFC5764], and no explicit negotiation is required in order to indicate support and usage of the mechanism.

[RFC5764] does not describe how to identify different protocols transported on DTLS, only how to identify the DTLS protocol itself. If multiple protocols are transported on DTLS, there MUST exist a specification describing a mechanism for identifying each individual protocol. In addition, if a received DTLS packet can be associated with more than one "m=" line, there MUST exist a specification which describes a mechanism for associating the received DTLS packet with the correct "m=" line.

[Section 10.2] describes how to associate a received (S)RTP packet with the correct "m=" line.

10. RTP Considerations

10.1. Single RTP Session

10.1.1. General

All RTP-based media within a single BUNDLE group belong to a single RTP session [RFC3550]. Disjoint BUNDLE groups will form multiple RTP sessions, one per BUNDLE group.

Since a single RTP session is used for each bundle group, all "m=" lines representing RTP-based media in a bundle group will share a single SSRC numbering space [RFC3550].

The following rules and restrictions apply for a single RTP session:

- A specific payload type value can be used in multiple bundled "m=" lines if each codec associated with the payload type number shares an identical codec configuration [Section 10.1.2].

- The proto value in each bundled RTP-based "m=" line MUST be identical (e.g. RTP/AVPF).

- The RTP MID header extension MUST be enabled, by associating an SDP 'extmap' attribute [RFC5285], with a 'urn:ietf:params:rtp-hdrext:sdes:mid' URI value, with each bundled RTP-based "m=" line in every offer and answer.

- A given SSRC MUST NOT transmit RTP packets using payload types that originate from different bundled "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 fail to function. Even if done in proper sequence this causes RTP Timestamp rate switching issues [RFC7160]. However, once an SSRC has left the RTP session (by sending an RTCP BYE packet), that SSRC value can later be reused by another source(possible associated with a different bundled "m=" line.

10.1.2. Payload Type (PT) Value Reuse

Multiple bundled "m=" lines might represent RTP based media. As all RTP based media associated with a BUNDLE group belong to the same RTP session, in order for a given payload type value to be used inside more than one bundled "m=" line, all codecs associated with the payload type number MUST share an identical codec configuration. This means that the codecs MUST share the same media type, encoding name, clock rate and any parameter that can affect the codec configuration and packetization. [I-D.mmusic-sdp-mux-attributes] lists SDP attributes, whose attribute values must be identical for all codecs that use the same payload type value.

10.2. Associating RTP/RTCP Packets With Correct SDP Media Description

There are multiple mechanisms that can be used by an endpoint in order to associate received RTP/RTCP packets with a bundled "m=" line. Such mechanisms include using the payload type value carried inside the RTP packets, the SSRC values carried inside the RTP packets, and other "m=" line specific information carried inside the RTP packets.

As all RTP/RTCP packets associated with a BUNDLE group are received (and sent) using single address:port combinations, the local address:port combination cannot be used to associate received RTP packets with the correct "m=" line.

As described in [Section 10.1.2], the same payload type value might be used inside RTP packets described by multiple "m=" lines. In such cases, the payload type value cannot be used to associate received RTP packets with the correct "m=" line.

An offerer and answerer can in an offer and answer inform each other which SSRC values they will use inside sent RTP/RTCP packets, by associating one or more SDP 'ssrc' attributes [RFC5576] with each bundled "m=" line which contains a payload type value that is also used inside another bundled "m=" line. As the SSRC values will be carried inside the RTP/RTCP packets, the offerer and answerer can then use that information to associate received RTP packets with the correct "m=" line. However, an offerer will not know which SSRC values the answerer will use until it has received the answer providing that information. Due to this, before the offerer has received the answer, the offerer will not be able to associate received RTP/RTCP packets with the correct "m=" line using the SSRC values.

In order for an offerer and answerer to always be able to associate received RTP and RTCP packets with the correct "m=" line, an offerer and answerer using the BUNDLE extension MUST support the mechanism defined in Section 13, where the remote endpoint inserts the identification-tag associated with an "m=" line in RTP and RTCP packets associated with that "m=" line.

10.3. RTP/RTCP Multiplexing

10.3.1. General

When a BUNDLE group, which contains RTP based media, is created, the offerer and answerer MUST negotiate whether to enable RTP/RTCP multiplexing for the RTP based media associated with the BUNDLE group [RFC5761].

If RTP/RTCP multiplexing is not enabled, separate address:port combinations will be used for receiving (and sending) the RTP packets and the RTCP packets.

10.3.2. SDP Offer/Answer Procedures

10.3.2.1. General

This section describes how an offerer and answerer can use the SDP 'rtcp-mux' attribute [RFC5761] and the SDP 'rtcp' attribute [RFC3605] to negotiate usage of RTP/RTCP multiplexing for RTP based media associated with a BUNDLE group.

10.3.2.2. Generating the Initial SDP Offer

When an offerer generates an initial offer, if the offerer wants to negotiate usage of RTP/RTCP multiplexing within a BUNDLE group, the offerer MUST associate an SDP 'rtcp-mux' attribute [RFC5761] with

each bundled RTP-based "m=" line (including any bundle-only "m=" line) in the offer.

If the offerer does not want to negotiate usage of RTP/RTCP multiplexing, it MUST NOT associate an SDP 'rtcp-mux' attribute with any bundled "m=" line in the offer.

In addition, the offerer can associate an SDP 'rtcp' attribute [RFC3605] with one or more bundled RTP-based "m=" lines (including any bundle-only "m=" line) in the offer, in order to provide a port for receiving RTCP packets (if the answerer does not accept usage of RTP/RTCP multiplexing, or if the offerer does not want to negotiate usage of RTP/RTCP multiplexing).

In the initial offer, the IP address and port combination for RTCP MUST be unique in each bundled RTP-based "m=" line, similar to RTP.

NOTE: In case the offer wants to receive RTCP packets on the next higher port value, the SDP 'rtcp' attribute is not needed.

10.3.2.3. Generating the SDP Answer

When an answerer generates an answer, if the offerer indicated support of RTP/RTCP multiplexing [RFC5761] within a BUNDLE group in the associated offer, the answerer MUST either accept or reject the usage of RTP/RTCP multiplexing for the whole BUNDLE group in the answer.

If the answerer accepts the usage of RTP/RTCP multiplexing within the BUNDLE group, it MUST associate an SDP 'rtcp-mux' attribute with each bundled RTP-based "m=" line in the answer. The answerer MUST NOT associate an SDP 'rtcp' attribute with any bundled "m=" line in the answer. The answerer will use the port value of the selected offerer BUNDLE address for sending RTP and RTCP packets associated with each RTP-based bundled "m=" line towards the offerer.

If the answerer does not accept the usage of RTP/RTCP multiplexing within the BUNDLE group, it MUST NOT associate an SDP 'rtcp-mux' attribute with any bundled "m=" line in the answer. The answerer will use the RTP and RTCP port values associated with the selected offerer BUNDLE address for sending RTP and RTCP packets associated with each RTP-based bundled "m=" line towards the offerer.

In addition, if the answerer rejects the usage of RTP/RTCP multiplexing within the BUNDLE group, it MAY associate an SDP 'rtcp' attribute, with identical attribute values, with each RTP-based bundled "m=" line in the answer, in order to provide a port value for receiving RTCP packets from the offerer.

NOTE: In case the answerer wants to receive RTCP packets on the next higher port value, the SDP 'rtcp' attribute is not needed.

If the usage of RTP/RTCP multiplexing within a BUNDLE group has been negotiated in a previous offer/answer transaction, and if the offerer indicates that it wants to continue using RTP/RTCP multiplexing in a subsequent offer, the answerer MUST associate an SDP 'rtcp-mux' attribute with each bundled "m=" line in the answer. I.e. the answerer MUST NOT disable the usage of RTP/RTCP multiplexing.

If the usage of RTP/RTCP multiplexing within a BUNDLE group has not been negotiated in a previous offer/answer transaction, and if the offerer indicates that it wants to use RTP/RTCP multiplexing in a subsequent offer, the answerer either accepts or rejects the usage, using the procedures above.

10.3.2.4. Offerer Processing of the SDP Answer

When an offerer receives an answer, if the answerer has accepted the usage of RTP/RTCP multiplexing (see Section 10.3.2.3), the answerer follows the procedures for RTP/RTCP multiplexing defined in [RFC5761]. The offerer will use the port value associated with the answerer BUNDLE address for sending RTP and RTCP packets associated with each RTP-based bundled "m=" line towards the answerer.

If the answerer did not accept the usage of RTP/RTCP multiplexing (see Section 10.3.2.3), the offerer will use separate address:port combinations for sending RTP and RTCP packets towards the answerer. If the answerer associated an SDP 'rtcp' attribute with the "m=" line representing the answerer BUNDLE address, the offerer will use the attribute port value for sending RTCP packets associated with each bundled RTP-based "m=" line towards the answerer. Otherwise the offerer will use the next higher port value associated with the answerer BUNDLE address for sending RTCP packets towards the answerer.

10.3.2.5. Modifying the Session

When an offerer generates a subsequent offer, if it wants to negotiate the usage of RTP/RTCP multiplexing within a BUNDLE group, or if it wants to continue the use of previously negotiated RTP/RTCP multiplexing, it MUST associate an SDP 'rtcp-mux' attribute with each RTP-based bundled "m=" line (including any bundled "m=" line that the offerer wants to add to the BUNDLE group), unless the offerer wants to disable or remove the "m=" line from the BUNDLE group.

If the offerer does not want to negotiate the usage of RTP/RTCP multiplexing within the BUNDLE group, or if it wants to disable

previously negotiated usage of RTP/RTCP multiplexing, it MUST NOT associate an SDP 'rtcp-mux' and attribute with any bundled "m=" line in the subsequent offer.

In addition, if the offerer does not indicate support of RTP/RTCP multiplexing within the subsequent offer, it MAY associate an SDP 'rtcp' attribute, with identical attribute values, with each RTP-based bundled "m=" line (including any bundled "m=" line that the offerer wants to add to the BUNDLE group), in order to provide a port for receiving RTCP packets.

NOTE: It is RECOMMENDED that, once the usage of RTP/RTCP multiplexing has been negotiated within a BUNDLE group, that the usage is not disabled. Disabling RTP/RTCP multiplexing means that the offerer and answerer need to reserve new ports, to be used for sending and receiving RTCP packets. Similar, if the usage of a specific RTCP port has been negotiated within a BUNDLE group, it is RECOMMENDED that the port value is not modified.

11. ICE Considerations

11.1. General

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

The procedures defined in [RFC5245] also apply to usage of ICE with BUNDLE, with the following exception:

- When BUNDLE addresses for a BUNDLE group have been selected for both endpoints, ICE connectivity checks and keep-alives only need to be performed for the whole BUNDLE group, instead of per bundled "m=" line.

Support and usage of ICE mechanism together with the BUNDLE extension is OPTIONAL.

11.2. SDP Offer/Answer Procedures

11.2.1. General

When an offerer assigns a unique address to a bundled "m=" line (excluding any bundle-only "m=" line), it MUST also associate unique ICE candidates [RFC5245] to the "m=" line.

An offerer MUST NOT assign ICE candidates to a bundle-only "m=" line with a zero port value.

NOTE: The bundle-only "m=" line, if accepted by the answerer, will inherit the candidates associated with the selected offerer BUNDLE address. An answerer that does not support BUNDLE would not accept a bundle-only "m=" line.

When an offerer or answerer assigns a shared address (i.e. a previously selected BUNDLE address) to one or more bundled "m=" lines, it MUST associate identical ICE candidates (referred to as shared ICE candidates) to each of those "m=" lines.

11.2.2. Generating the Initial SDP Offer

When an offerer generates an initial offer, it assigns unique or shared ICE candidates to the bundled "m=" lines, according to Section 11.1.

11.2.3. Generating the SDP Answer

When an answerer generates an answer, which contains a BUNDLE group, the answerer MUST assign shared ICE candidates to each bundled "m=" line (including "m=" lines that were indicated as bundle-only in the associated offer) in the answer.

11.2.4. Offerer Processing of the SDP Answer

When an offerer receives an answer, if the answerer supports and uses the ICE mechanism and the BUNDLE extension, the offerer MUST assign the same ICE candidates, associated with the "m=" line representing the offerer BUNDLE address (selected by the answerer), to each bundled "m=" line.

11.2.5. Modifying the Session

When an offerer generates a subsequent offer, it assigns unique or shared ICE candidates to the bundled "m=" lines, according to (Section 11.1).

12. Update to RFC 3264

12.1. General

This section replaces the text of the following sections of RFC 3264:

- Section 5.1 (Unicast Streams).
- Section 8.2 (Removing a Media Stream).
- Section 8.4 (Putting a Unicast Media Stream on Hold).

12.2. Original text of section 5.1 (2nd paragraph) of RFC 3264

For recvonly and sendrecv streams, the port number and address in the offer indicate where the offerer would like to receive the media stream. For sendonly RTP streams, the address and port number indirectly indicate where the offerer wants to receive RTCP reports. Unless there is an explicit indication otherwise, reports are sent to the port number one higher than the number indicated. The IP address and port present in the offer indicate nothing about the source IP address and source port of RTP and RTCP packets that will be sent by the offerer. A port number of zero in the offer indicates that the stream is offered but MUST NOT be used. This has no useful semantics in an initial offer, but is allowed for reasons of completeness, since the answer can contain a zero port indicating a rejected stream (Section 6). Furthermore, existing streams can be terminated by setting the port to zero (Section 8). In general, a port number of zero indicates that the media stream is not wanted.

12.3. New text replacing section 5.1 (2nd paragraph) of RFC 3264

For recvonly and sendrecv streams, the port number and address in the offer indicate where the offerer would like to receive the media stream. For sendonly RTP streams, the address and port number indirectly indicate where the offerer wants to receive RTCP reports. Unless there is an explicit indication otherwise, reports are sent to the port number one higher than the number indicated. The IP address and port present in the offer indicate nothing about the source IP address and source port of RTP and RTCP packets that will be sent by the offerer. A port number of zero in the offer by default indicates that the stream is offered but MUST NOT be used, but an extension mechanism might specify different semantics for the usage of a zero port value. Furthermore, existing streams can be terminated by setting the port to zero (Section 8). In general, a port number of zero by default indicates that the media stream is not wanted.

12.4. Original text of section 8.2 (2nd paragraph) of RFC 3264

A stream that is offered with a port of zero MUST be marked with port zero in the answer. Like the offer, the answer MAY omit all attributes present previously, and MAY list just a single media format from amongst those in the offer.

12.5. New text replacing section 8.2 (2nd paragraph) of RFC 3264

A stream that is offered with a port of zero MUST by default be marked with port zero in the answer, unless an extension mechanism, which specifies semantics for the usage of a non-zero port value, is used. If the stream is marked with port zero in the answer, the

answer MAY omit all attributes present previously, and MAY list just a single media format from amongst those in the offer."

12.6. Original text of section 8.4 (6th paragraph) of RFC 3264

RFC 2543 [10] specified that placing a user on hold was accomplished by setting the connection address to 0.0.0.0. Its usage for putting a call on hold is no longer recommended, since it doesn't allow for RTCP to be used with held streams, doesn't work with IPv6, and breaks with connection oriented media. However, it can be useful in an initial offer when the offerer knows it wants to use a particular set of media streams and formats, but doesn't know the addresses and ports at the time of the offer. Of course, when used, the port number MUST NOT be zero, which would specify that the stream has been disabled. An agent MUST be capable of receiving SDP with a connection address of 0.0.0.0, in which case it means that neither RTP nor RTCP should be sent to the peer.

12.7. New text replacing section 8.4 (6th paragraph) of RFC 3264

RFC 2543 [10] specified that placing a user on hold was accomplished by setting the connection address to 0.0.0.0. Its usage for putting a call on hold is no longer recommended, since it doesn't allow for RTCP to be used with held streams, doesn't work with IPv6, and breaks with connection oriented media. However, it can be useful in an initial offer when the offerer knows it wants to use a particular set of media streams and formats, but doesn't know the addresses and ports at the time of the offer. Of course, when used, the port number MUST NOT be zero, if it would specify that the stream has been disabled. However, an extension mechanism might specify different semantics of the zero port number usage. An agent MUST be capable of receiving SDP with a connection address of 0.0.0.0, in which case it means that neither RTP nor RTCP should be sent to the peer.

13. RTP/RTCP extensions for identification-tag transport

13.1. General

SDP Offerers and Answerers [RFC3264] can associate identification-tags with "m=" lines within SDP Offers and Answers, using the procedures in [RFC5888]. Each identification-tag uniquely represents an "m=" line.

This section defines a new RTCP SDES item [RFC3550], 'MID', which is used to carry identification-tags within RTCP SDES packets. This section also defines a new RTP header extension [RFC5285], which is used to carry identification-tags in RTP packets.

The SDES item and RTP header extension make it possible for a receiver to associate received RTCP- and RTP packets with a specific "m=" line, to which the receiver has assigned an identification-tag, even if those "m=" lines are part of the same RTP session. The endpoint informs the remote endpoint about the identification-tag using the procedures in [RFC5888], and the remote endpoint then inserts the identification-tag in RTCP- and RTP packets sent towards the other endpoint.

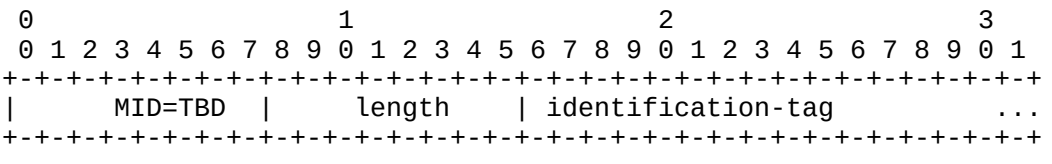
NOTE: This text above defines how identification-tags are carried in SDP Offers and Answers. The usage of other signalling protocols for carrying identification-tags is not prevented, but the usage of such protocols is outside the scope of this document.

[RFC3550] defines general procedures regarding the RTCP transmission interval. The RTCP MID SDES item SHOULD be sent in the first few RTCP packets sent on joining the session, and SHOULD be sent regularly thereafter. The exact number of RTCP packets in which this SDES item is sent is intentionally not specified here, as it will depend on the expected packet loss rate, the RTCP reporting interval, and the allowable overhead.

The RTP MID header extension SHOULD be included in some RTP packets at the start of the session and whenever the SSRC changes. It might also be useful to include the header extension in RTP packets that comprise random access points in the media (e.g., with video I-frames). The exact number of RTP packets in which this header extension is sent is intentionally not specified here, as it will depend on expected packet loss rate and loss patterns, the overhead the application can tolerate, and the importance of immediate receipt of the identification-tag.

For robustness purpose, endpoints need to be prepared for situations where the reception of the identification-tag is delayed, and SHOULD NOT terminate sessions in such cases, as the identification-tag is likely to arrive soon.

13.2. RTCP MID SDES Item



The identification-tag payload is UTF-8 encoded, as in SDP.

The identification-tag is not zero terminated.

[RFC EDITOR NOTE: Please replace TBD with the assigned SDES identifier value.]

13.3. RTP MID Header Extension

The payload, containing the identification-tag, of the RTP MID header extension element can be encoded using either the one-byte or two-byte header [RFC5285]. The identification-tag payload is UTF-8 encoded, as in SDP.

The identification-tag is not zero terminated. Note, that set of header extensions included in the packet needs to be padded to the next 32-bit boundary using zero bytes [RFC5285].

As the identification-tag is included in either an RTCP SDES item or an RTP header extension, or both, there should be some consideration about the packet expansion caused by the identification-tag. To avoid Maximum Transmission Unit (MTU) issues for the RTP packets, the header extension’s size needs to be taken into account when the encoding media.

It is recommended that the identification-tag is kept short. Due to the properties of the RTP header extension mechanism, when using the one-byte header, a tag that is 1-3 bytes will result in that a minimal number of 32-bit words are used for the RTP header extension, in case no other header extensions are included at the same time. Note, do take into account that some single characters when UTF-8 encoded will result in multiple octets.

14. IANA Considerations

14.1. New SDES item

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

[RFC EDITOR NOTE: Please replace TBD with the assigned SDES identifier value.]

This document adds the MID SDES item to the IANA "RTCP SDES item types" registry as follows:

Value:	TBD
Abbrev.:	MID
Name:	Media Identification
Reference:	RFCXXXX

14.2. New RTP Header Extension URI

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

This document defines a new extension URI in the RTP Compact Header Extensions subregistry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI:	urn:ietf:params:rtp-hdrext:sdes:mid
Description:	Media identification
Contact:	christer.holmberg@ericsson.com
Reference:	RFCXXXX

14.3. New SDP Attribute

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

This document defines a new SDP media-level attribute, 'bundle-only', according to the following data:

Attribute name:	bundle-only
Type of attribute:	media
Subject to charset:	No
Purpose:	Request a media description to be accepted in the answer only if kept within a BUNDLE group by the answerer.
Appropriate values:	N/A
Contact name:	Christer Holmberg
Contact e-mail:	christer.holmberg@ericsson.com
Reference:	RFCXXXX

15. Security Considerations

The security considerations defined in [RFC3264] and [RFC5888] apply to the BUNDLE extension. Bundle does not change which information flows over the network but only changes which ports that information

is flowing on and thus has very little impact on the security of the RTP sessions.

When the BUNDLE extension is used, a single set of security credentials might be used for all media streams associated with a BUNDLE group.

When the BUNDLE extension is used, the number of SSRC values within a single RTP session increases, which increases the risk of SSRC collision. [RFC4568] describes how SSRC collision may weaken SRTP and SRTCP encryption in certain situations.

16. Examples

16.1. Example: Bundle Address Selection

The example below shows:

- 1. An offer, in which the offerer assigns a unique address to each bundled "m=" line within the BUNDLE group.
- 2. An answer, in which the answerer selects the offerer BUNDLE address, and in which selects its own BUNDLE address (the answerer BUNDLE address) and assigns it each bundled "m=" line within the BUNDLE group.
- 3. A subsequent offer (BAS offer), which is used to perform a Bundle Address Synchronization (BAS).

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10002 RTP/AVP 31 32
b=AS:1000
a=mid:bar
```

```
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 20000 RTP/AVP 32
b=AS:1000
a=mid:bar
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
```

SDP Offer (3)

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
```

16.2. Example: BUNDLE Extension Rejected

The example below shows:

- 1. An offer, in which the offerer assigns a unique address to each bundled "m=" line within the BUNDLE group.
- 2. An answer, in which the answerer rejects the offered BUNDLE group, and assigns a unique addresses to each "m=" line (following normal RFC 3264 procedures).

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10002 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
```

SDP Answer (2)

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

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

The example below shows:

- 1. A subsequent offer (the BUNDLE group has been created as part of a previous offer/answer transaction), in which the offerer adds a new "m=" line, represented by the "zen" identification-tag, to a previously negotiated BUNDLE group, assigns a unique address to the added "m=" line, and assigns the previously selected offerer

BUNDLE address to each of the other bundled "m=" lines within the BUNDLE group.

- 2. An answer, in which the answerer assigns the answerer BUNDLE address to each bundled "m=" line (including the newly added "m=" line) within the BUNDLE group.

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

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar zen
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 20000 RTP/AVP 66
b=AS:1000
a=mid:zen
a=rtpmap:66 H261/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar zen
m=audio 20000 RTP/AVP 0
```

```
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 20000 RTP/AVP 32
b=AS:1000
a=mid:bar
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 20000 RTP/AVP 66
b=AS:1000
a=mid:zen
a=rtpmap:66 H261/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
```

SDP Offer (3)

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar zen
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10000 RTP/AVP 66
b=AS:1000
a=mid:zen
a=rtpmap:66 H261/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
```

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

The example below shows:

- 1. A subsequent offer (the BUNDLE group has been created as part of a previous offer/answer transaction), in which the offerer moves a bundled "m=" line out of a BUNDLE group, assigns a unique address to the moved "m=" line, and assigns the offerer BUNDLE address to each other bundled "m=" line within the BUNDLE group.

- 2. An answer, in which the answerer moves the "m=" line out of the BUNDLE group, assigns unique address to the moved "m=" line, and assigns the answerer BUNDLE address to each of the remaining bundled "m=" line within the BUNDLE group.

SDP Offer (1)

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 50000 RTP/AVP 66
b=AS:1000
a=mid:zen
a=rtpmap:66 H261/90000
```

SDP Answer (2)

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

```
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 20000 RTP/AVP 32
b=AS:1000
a=mid:bar
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 60000 RTP/AVP 66
b=AS:1000
a=mid:zen
a=rtpmap:66 H261/90000
```

16.5. Example: Offerer Disables A Media Description Within A BUNDLE Group

The example below shows:

- 1. A subsequent offer (the BUNDLE group has been created as part of a previous offer/answer transaction), in which the offerer disables a bundled "m=" line within BUNDLE group, assigns a zero port number to the disabled "m=" line, and assigns the offerer BUNDLE address to each of the other bundled "m=" lines within the BUNDLE group.

- 2. An answer, in which the answerer moves the disabled "m=" line out of the BUNDLE group, assigns a zero port value to the disabled "m=" line, and assigns the answerer BUNDLE address to each of the remaining bundled "m=" line within the BUNDLE group.

SDP Offer (1)

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

```
a=mid:foo
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 10000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 0 RTP/AVP 66
a=mid:zen
a=rtpmap:66 H261/90000
```

SDP Answer (2)

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 20000 RTP/AVP 32
b=AS:1000
a=mid:bar
a=rtpmap:32 MPV/90000
a=extmap 1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 0 RTP/AVP 66
a=mid:zen
a=rtpmap:66 H261/90000
```

17. Acknowledgements

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

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

Thanks to Paul Kyzivat, Martin Thomson, Flemming Andreasen, Thomas Stach and Ari Keraenen for taking the time to read the text along the way, and providing useful feedback.

18. Change Log

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

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-17

- - Editorial changes based on comments from Magnus Westerlund.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-16

- - Modification of RTP/RTCP multiplexing section, based on comments from Magnus Westerlund.
- - Reference updates.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-15

- - Editorial fix.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-14

- - Editorial changes.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-13

- Changes to allow a new suggested offerer BUNDLE address to be assigned to each bundled m- line.
- Changes based on WGLC comments from Paul Kyzivat
- - Editorial fixes

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-12

- Usage of SDP ’extmap’ attribute added
- SDP ’bundle-only’ attribute scoped with "m=" lines with a zero port value
- Changes based on WGLC comments from Thomas Stach

- o - ICE candidates not assigned to bundle-only m- lines with a zero port value
- o - Editorial changes
- o Changes based on WGLC comments from Colin Perkins
- o - Editorial changes:
- o -- "RTP SDES item" -> "RTCP SDES item"
- o -- "RTP MID SDES item" -> "RTCP MID SDES item"
- o - Changes in section 10.1.1:
- o -- "SHOULD NOT" -> "MUST NOT"
- o -- Additional text added to the Note
- o - Change to section 13.2:
- o -- Clarify that mid value is not zero terminated
- o - Change to section 13.3:
- o -- Clarify that mid value is not zero terminated
- o -- Clarify padding
- o Changes based on WGLC comments from Paul Kyzivat
- o - Editorial changes:
- o Changes based on WGLC comments from Jonathan Lennox
- o - Editorial changes:
- o - Defintion of SDP bundle-only attribute alligned with structure in 4566bis draft

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-11

- o Editorial corrections based on comments from Harald Alvestrand.
- o Editorial corrections based on comments from Cullen Jennings.
- o Reference update (RFC 7160).

- Clarification about RTCP packet sending when RTP/RTCP multiplexing is not used (http://www.ietf.org/mail-archive/web/mmusic/current/msg13765.html).
- Additional text added to the Security Considerations.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-10

- SDP bundle-only attribute added to IANA Considerations.
- SDES item and RTP header extension added to Abstract and Introduction.
- Modification to text updating section 8.2 of RFC 3264.
- Reference corrections.
- Editorial corrections.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-09

- Terminology change: "bundle-only attribute assigned to m= line" to "bundle-only attribute associated with m= line".
- Editorial corrections.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-08

- Editorial corrections.
- - "of"->"if" (8.3.2.5).
- - "optional"->"OPTIONAL" (9.1).
- - Syntax/ABNF for ’bundle-only’ attribute added.
- - SDP Offer/Answer sections merged.
- - ’Request new offerer BUNDLE address’ section added

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-07

- OPEN ISSUE regarding Receiver-ID closed.
- - RTP MID SDES Item.
- - RTP MID Header Extension.

- OPEN ISSUE regarding insertion of SDP ’rtcp’ attribute in answers closed.
- - Indicating that, when rtcp-mux is used, the answerer MUST NOT include an ’rtcp’ attribute in the answer, based on the procedures in section 5.1.3 of RFC 5761.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-06

- Draft title changed.
- Added "SDP" to section names containing "Offer" or "Answer".
- Editorial fixes based on comments from Paul Kyzivat (http://www.ietf.org/mail-archive/web/mmusic/current/msg13314.html).
- Editorial fixed based on comments from Colin Perkins (http://www.ietf.org/mail-archive/web/mmusic/current/msg13318.html).
- - Removed text about extending BUNDLE to allow multiple RTP sessions within a BUNDLE group.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-05

- Major re-structure of SDP Offer/Answer sections, to align with RFC 3264 structure.
- Additional definitions added.
- - Shared address.
- - Bundled "m=" line.
- - Bundle-only "m=" line.
- - Offerer suggested BUNDLE mid.
- - Answerer selected BUNDLE mid.
- Q6 Closed (IETF#88): An Offerer MUST NOT assign a shared address to multiple "m=" lines until it has received an SDP Answer indicating support of the BUNDLE extension.
- Q8 Closed (IETF#88): An Offerer can, before it knows whether the Answerer supports the BUNDLE extension, assign a zero port value to a ’bundle-only’ "m=" line.

- SDP 'bundle-only' attribute section added.
- Connection data nettype/addrtype restrictions added.
- RFC 3264 update section added.
- Indicating that a specific payload type value can be used in multiple "m=" lines, if the value represents the same codec configuration in each "m=" line.

Changes from draft-ietf-mmusic-sdp-bundle-negotiation-04

- Updated Offerer procedures (http://www.ietf.org/mail-archive/web/mmusic/current/msg12293.html).
- Updated Answerer procedures (http://www.ietf.org/mail-archive/web/mmusic/current/msg12333.html).
- Usage of SDP 'bundle-only' attribute added.
- Reference to Trickle ICE document added.

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

- 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.
- Cullen Jennings added as co-author.

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

- No changes. New version due to expiration.

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

- No changes. New version due to expiration.

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

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

- Added reference to RFC 3550.

19. References

19.1. 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.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

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

[I-D.mmusic-sdp-mux-attributes] Nandakumar, S., "A Framework for SDP Attributes when Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-08 (work in progress), January 2015.

19.2. 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.

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, 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.

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

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC7160] Petit-Huguenin, M. and G. Zorn, "Support for Multiple Clock Rates in an RTP Session", RFC 7160, April 2014.

[I-D.ietf-mmusic-trickle-ice] Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", draft-ietf-mmusic-trickle-ice-02 (work in progress), January 2015.

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 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 address:port combination 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 values.

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

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

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

A.2. UA Interoperability

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

SDP Offer

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

SDP Answer

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

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

symmetric RTP and ICE still use a SDP data structure where SDP with "m=" lines with the same port such as:

SDP Offer

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

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

A.3. Usage of port number value zero

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

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

A.4. B2BUA And Proxy Interoperability

Some back to back user agents may be configured in a mode where if the incoming call leg contains an SDP attribute the B2BUA does not understand, the B2BUS still generates that SDP attribute in the Offer for the outgoing call leg. Consider an B2BUA that did not understand the SDP "rtcp" attribute, defined in RFC 3605, yet acted this way. Further assume that the B2BUA was configured to tear down any call

where it did not see any RTCP for 5 minutes. In this cases, if the B2BUA received an Offer like:

SDP Offer

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

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

A.4.1. Traffic Policing

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

A.4.2. Bandwidth Allocation

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

A.5. Candidate Gathering

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

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

Authors' Addresses

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Harald Tveit Alvestrand
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

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

Email: fluffy@iii.ca

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 24, 2015

B. Burman
M. Westerlund
Ericsson
S. Nandakumar
M. Zanaty
Cisco

January 20, 2015

Using Simulcast in SDP and RTP Sessions
draft-ietf-mmusic-sdp-simulcast-00

Abstract

In some application scenarios it may be desirable to send multiple differently encoded versions of the same media source in independent RTP streams. This is called simulcast. This document discusses the best way of accomplishing simulcast in RTP and how to signal it in SDP. A solution is defined by making an extension to SDP, and using RTP/RTCP identification methods to relate RTP streams belonging to the same media source. The SDP extension consists a new media level SDP attribute that express capability to send and/or receive simulcast RTP streams. One part of the RTP/RTCP identification method is included as a reference to a separate document, since it is useful also for other purposes.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 24, 2015.

Table of Contents

1. Introduction

Most of today's multiparty video conference solutions make use of centralized servers to reduce the bandwidth and CPU consumption in the endpoints. Those servers receive RTP streams from each participant and send some suitable set of possibly modified RTP streams to the rest of the participants, which usually have heterogeneous capabilities (screen size, CPU, bandwidth, codec, etc). One of the biggest issues is how to perform RTP stream adaptation to different participants' constraints with the minimum possible impact on both video quality and server performance.

Simulcast is defined in this memo as the act of simultaneously sending multiple different encoded streams of the same media source, e.g. the same video source encoded with different video encoder types or image resolutions. This can be done in several ways and for different purposes. This document focuses on the case where it is desirable to provide a media source as multiple encoded streams over RTP [RFC3550] towards an intermediary so that the intermediary can provide the wanted functionality by selecting which RTP stream to forward to other participants in the session, and more specifically how the identification and grouping of the involved RTP streams are done. From an RTP perspective, simulcast is a specific application of the aspects discussed in RTP Multiplexing Guidelines [I-D.ietf-avtcore-multiplex-guidelines].

The purpose of this document is to describe a few scenarios where it is motivated to use simulcast, and propose a suitable solution for SDP signaling and performing RTP simulcast.

2. Definitions

2.1. Terminology

This document makes use of the terminology defined in RTP Taxonomy [I-D.ietf-avtext-rtp-grouping-taxonomy], RTP Topology [RFC5117] and RTP Topologies Update [I-D.ietf-avtcore-rtp-topologies-update]. In addition, the following terms are used:

RTP Mixer: An RTP middle node, defined in [RFC5117] (Section 3.4: Topo-Mixer), further elaborated and extended with other topologies in [I-D.ietf-avtcore-rtp-topologies-update] (Section 3.6 to 3.9).

RTP Switch: A common short term for the terms "switching RTP mixer", "source projecting middlebox", and "video switching MCU" as discussed in [I-D.ietf-avtcore-rtp-topologies-update].

Simulcast version: One encoded stream from the set of encoded streams that constitutes the simulcast for a single media source.

Simulcast version alternative: One encoded stream being encoded in one of possibly multiple alternative ways to create a simulcast version.

2.2. Requirements Language

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

3. Use Cases

Many use cases of simulcast as described in this document relate to a multi-party communication session where one or more central nodes are used to adapt the view of the communication session towards individual participants, and facilitate the media transport between participants. Thus, these cases targets the RTP Mixer type of topology.

There are two principle approaches for an RTP Mixer to provide this adapted view of the communication session to each receiving participant:

- Transcoding (decoding and re-encoding) received RTP streams with characteristics adapted to each receiving participant. This often include mixing or composition of media sources from multiple participants into a mixed media source originated by the RTP Mixer. The main advantage of this approach is that it achieves close to optimal adaptation to individual receiving participants. The main disadvantages are that it can be very computationally expensive to the RTP Mixer and typically also degrades media Quality of Experience (QoE) such as end-to-end delay for the receiving participants.

- Switching a subset of all received RTP streams or sub-streams to each receiving participant, where the used subset is typically specific to each receiving participant. The main advantages of this approach are that it is computationally cheap to the RTP Mixer and it has very limited impact on media QoE. The main disadvantage is that it can be difficult to combine a subset of received RTP streams into a perfect fit to the resource situation of a receiving participant.

The use of simulcast relates to the latter approach, where it is more important to reduce the load on the RTP Mixer and/or minimize QoE impact than to achieve an optimal adaptation of resource usage.

A multicast/broadcast case where the receivers themselves selects the most appropriate simulcast version and tune in to the right media transport to receive that version is also considered (Section 3.3) . This enables large, heterogeneous receiver populations, when it comes to capabilities and the use of network path bandwidth resources.

3.1. Reaching a Diverse Set of Receivers

The media sources provided by a sending participant potentially need to reach several receiving participants that differ in terms of available resources. The receiver resources that typically differ include, but are not limited to:

Codec: This includes codec type (such as SDP MIME type) and can include codec configuration options (e.g. SDP fmtp parameters). A couple of codec resources that differ only in codec configuration will be "different" if they are somehow not "compatible", like if they differ in video codec profile, or the transport packetization configuration.

Sampling: This relates to how the media source is sampled, in spatial as well as in temporal domain. For video streams, spatial sampling affects image resolution and temporal sampling affects video frame rate. For audio, spatial sampling relates to the number of audio channels and temporal sampling affects audio bandwidth. This may be used to suit different rendering capabilities or needs at the receiving endpoints, as well as a method to achieve different transport capabilities, bitrates and eventually QoE by controlling the amount of source data.

Bitrate: This relates to the amount of bits spent per second to transmit the media source as an RTP stream, which typically also affects the Quality of Experience (QoE) for the receiving user.

Letting the sending participant create a simulcast of a few differently configured RTP streams per media source can be a good tradeoff when using an RTP switch as middlebox, instead of sending a single RTP stream and using an RTP mixer to create individual transcodings to each receiving participant.

This requires that the receiving participants can be categorized in terms of available resources and that the sending participant can choose a matching configuration for a single RTP stream per category and media source.

For example, assume for simplicity a set of receiving participants that differ only in that some have support to receive Codec A, and the others have support to receive Codec B. Further assume that the sending participant can send both Codec A and B. It can then reach all receivers by creating two simulcasted RTP streams from each media source; one for Codec A and one for Codec B.

In another simple example, a set of receiving participants differ only in screen resolution; some are able to display video with at most 360p resolution and some support 720p resolution. A sending participant can then reach all receivers by creating a simulcast of RTP streams with 360p and 720p resolution for each sent video media source.

In more elaborate cases, the receiving participants differ both in available sampling and bitrate, and maybe also codec, and it is up to the RTP switch to find a good trade-off in which simulcasted stream to choose for each intended receiver. It is also the responsibility of the RTP switch to negotiate a good fit of simulcast streams with the sending participant.

The maximum number of simulcasted RTP streams that can be sent is mainly limited by the amount of processing and uplink network resources available to the sending participant.

3.2. Application Specific Media Source Handling

The application logic that controls the communication session may include special handling of some media sources. It is for example commonly the case that the media from a sending participant is not sent back to itself.

It is also common that a currently active speaker participant is shown in larger size or higher quality than other participants (the sampling or bitrate aspects of Section 3.1). Not sending the active speaker media back to itself means there is some other participant’s media that instead has to receive special handling towards the active speaker; typically the previous active speaker. This way, the previously active speaker is needed both in larger size (to current active speaker) and in small size (to the rest of the participants), which can be solved with a simulcast from the previously active speaker to the RTP switch.

3.3. Receiver Adaptation in Multicast/Broadcast

When using broadcast or multicast technology to distribute real-time media streams to large populations of receivers, there can still be

significant heterogeneity among the receiver population. This can depend on several factors:

Network Bandwidth: The network paths to individual receivers will have variations in the bandwidth, thus putting different limits on the supported bit-rates that can be received.

Endpoint Capabilities: The end point's hardware and software can have varying capabilities in relation to screen resolution, decoding capabilities, and supported media codecs.

To handle these variations, a transmitter of real-time media may want to apply simulcast to a media source and provide it as a set of different encoded streams, enabling the receivers to select the best fit from this set themselves. The end point capabilities will usually result in a single initial choice. However, the network bandwidth can vary over time, which requires a client to continuously monitor its reception to determine if the received RTP streams still fit within the available bandwidth. If not, another set of encoded streams from the ones offered in the simulcast will have to be chosen.

When using IP multicast, the level of granularity that the receiver can select from is decided by its ability to choose different multicast addresses. Thus, different simulcast versions need to be put on different media transports using different multicast addresses. If these simulcast versions are described using SDP, they need to be part of different SDP media descriptions, as SDP binds to transport on media description level.

3.4. Receiver Media Source Preferences

The application logic that controls the communication session may allow receiving participants to apply preferences to the characteristics of the RTP stream they receive, for example in terms of the aspects listed in Section 3.1. Sending a simulcast of RTP streams is one way of accommodating receivers with conflicting or otherwise incompatible preferences.

4. Requirements

The following requirements need to be met to support the use cases in previous sections:

REQ-1: Identification. It must be possible to identify a set of simulcasted RTP streams as originating from the same media source:

REQ-1.1: In SDP signaling.

REQ-1.2: On RTP/RTCP level.

REQ-2: Transport usage. The solution must work when using:

REQ-2.1: Legacy SDP with separate media transports per SDP media description.

REQ-2.2: Bundled SDP media descriptions.

REQ-3: Capability negotiation. It must be possible that:

REQ-3.1: Sender can express capability of sending simulcast.

REQ-3.2: Receiver can express capability of receiving simulcast.

REQ-3.3: Sender can express maximum number of simulcast versions that can be provided.

REQ-3.4: Receiver can express maximum number of simulcast versions that can be received.

REQ-3.5: Sender can detail the characteristics of the simulcast versions that can be provided.

REQ-3.6: Receiver can detail the characteristics of the simulcast versions that it prefers to receive.

REQ-4: Distinguishing features. It must be possible to have different simulcast versions use different codec parameters, as can be expressed by SDP format values and RTP payload types.

REQ-5: Compatibility. It must be possible to use simulcast in combination with other RTP mechanisms that generate additional RTP streams:

REQ-5.1: RTP Retransmission [RFC4588].

REQ-5.2: RTP Forward Error Correction [RFC5109].

REQ-5.3: Related payload types such as audio Comfort Noise and/or DTMF.

REQ-6: Interoperability. The solution must be possible to use in:

REQ-6.1: Interworking with non-simulcast legacy clients using a single media source per media type.

REQ-6.2: WebRTC "Unified Plan" environment with a single media source per SDP media description.

5. Proposed Solution Overview

The proposed solution consists of signaling simulcast capability and configurations in SDP [RFC4566]:

- An offer or answer can contain a number of simulcast versions, separate for send and receive directions.
- An offer or answer can contain multiple, alternative simulcast versions in the same fashion as multiple, alternative codecs can be offered in a media description.
- Currently, a single media source per SDP media description is assumed, which makes the solution work in an Unified Plan [I-D.roach-mmusic-unified-plan] context (although different from what is currently defined there), both with and without BUNDLE grouping.
- The codec configuration for each simulcast version is expressed in terms of existing SDP formats (and typically RTP payload types). Some codecs may rely on codec configuration based on general attributes that apply for all formats within a media description, and which could thus not be used to separate different simulcast versions. This memo makes no attempt to address such shortcomings, but if needed instead encourages that a separate, general mechanism is defined for that purpose.
- It is possible, but not required to use source-specific signaling [RFC5576] with the proposed solution.

6. Proposed Solution

This section further details the signaling solution outlined above (Section 5).

6.1. Simulcast Capability

Simulcast capability is expressed as a new media level SDP attribute, "a=simulcast". For each desired direction (send/recv/sendrecv), the simulcast attribute defines a list of simulcast versions (separated by semicolons), each of which is a list of alternative RTP payload types (separated by commas) for that simulcast version. The meaning of the attribute on SDP session level is undefined and MUST NOT be used. There MUST be at most one "a=simulcast" attribute per media description. The ABNF [RFC5234] for this attribute is:

```
simulcast-attribute = "a=simulcast" 1*3( WSP sc-dir-list )
sc-dir-list         = sc-dir WSP sc-fmt-list *( ";" sc-fmt-list )
sc-dir              = "send" / "recv" / "sendrecv"
sc-fmt-list         = sc-fmt *( "," sc-fmt )
sc-fmt              = fmt
; WSP defined in [RFC5234]
; fmt defined in [RFC4566]
```

Figure 1: ABNF for Simulcast

There are separate and independent sets of parameters for simulcast in send and receive directions. When listing multiple directions, each direction MUST NOT occur more than once.

Attribute parameters are grouped by direction and consist of a listing of SDP format tokens (usually corresponding to RTP payload types), which describe the simulcast versions to be used. The number of (non-alternative, see below) formats in the list sets a limit to the number of supported simulcast versions in that direction. The order of the listed simulcast versions in the "send" direction is not significant. The order of the listed simulcast versions in the "recv" direction expresses a preference which simulcast versions that are preferred, with the leftmost being most preferred, if the number of actually sent simulcast versions have to be reduced for some reason.

Formats that have explicit dependencies [RFC5583] to other formats (even in the same media description) MAY be listed as different simulcast versions.

Alternative simulcast versions MAY be specified as part of the attribute parameters by expressing each simulcast version format as a comma-separated list of alternative values. In this case, all combinations of those alternatives MUST be supported. The order of the alternatives within a simulcast version is not significant; codec preference is expressed by format type ordering on the m-line, using regular SDP rules.

A simulcast version can use a codec defined such that the same RTP SSRC can change RTP payload type multiple times during a session, possibly even on a per-packet basis. A typical example can be a speech codec that makes use of Comfort Noise [RFC3389] and/or DTMF [RFC4733] formats. In those cases, such "related" formats MUST NOT be listed explicitly in the attribute parameters, since they are not strictly simulcast versions of the media source, but rather a specific way of generating the RTP stream of a single simulcast version with varying RTP payload type. Instead, only a single codec

format MUST be used per simulcast version or simulcast version alternative (if there are such). The codec format SHOULD be the codec most relevant to the media description, if possible to identify, for example the audio codec rather than the DTMF. What codec format to choose in the case of switching between multiple equally "important" formats is left open, but it is assumed that in the presence of such strong relation it does not matter which is chosen.

Use of the redundant audio data [RFC2198] format could be seen as a form of simulcast for loss protection purposes, but is not considered conflicting with the mechanisms described in this memo and MAY therefore be used as any other format. In this case the "red" format, rather than the carried formats, SHOULD be the one to list as a simulcast version on the "a=simulcast" line.

> Editor's note: Consider adding the possibility to put an RTP stream in "paused" state [I-D.ietf-avtext-rtp-stream-pause] from the beginning of the session, possibly starting it at a later point in time by applying RTP/RTCP level procedures from that specification.

6.1.1. Declarative Use

When used as a declarative media description, a=simulcast "recv" direction formats indicates the configured end point's required capability to recognize and receive a specified set of RTP streams as simulcast streams. In the same fashion, a=simulcast "send" direction requests the end point to send a specified set of RTP streams as simulcast streams. The "sendrecv" direction combines "send" and "recv" requirements, using the same format values for both.

If simulcast version alternatives are listed, it means that the configured end point MUST be prepared to receive any of the "recv" formats, and MAY send any of the "send" formats for that simulcast version.

6.1.2. Offer/Answer Use

An offerer wanting to use simulcast SHALL include the "a=simulcast" attribute in the offer. An offerer that receives an answer without "a=simulcast" MUST NOT use simulcast towards the answerer. An offerer that receives an answer with "a=simulcast" not listing a direction or without any formats in a specified direction MUST NOT use simulcast in that direction.

An answerer that does not understand the concept of simulcast will also not know the attribute and will remove it in the SDP answer, as

defined in existing SDP Offer/Answer [RFC3264] procedures. An answerer that does understand the attribute and that wants to support simulcast in an indicated direction SHALL reverse directionality of the unidirectional direction parameters; "send" becomes "recv" and vice versa, and include it in the answer. If the offered direction is "sendrecv", the answerer MAY keep it, but MAY also change it to "send" or "recv" to indicate that it is only interested in simulcast for a single direction. Note that, like all other use of SDP format tags for the send direction in Offer/Answer, format tags related to the simulcast send direction in an offer ("send" or "sendrecv") are placeholders that refer to information in the offer SDP, and the actual formats that will be used on the wire (including RTP Payload Format numbers) depends on information included in the SDP answer.

An offerer listing a set of receive simulcast versions and/or alternatives in the offer MUST be prepared to receive RTP streams for any of those simulcast versions and/or alternatives from the answerer.

An answerer that receives an offer with simulcast containing an "a=simulcast" attribute listing alternative formats for simulcast versions MAY keep all the alternatives in the answer, but it MAY also choose to remove any non-desirable alternatives per simulcast version in the answer. The answerer MUST NOT add any alternatives that were not present in the offer.

An answerer that receives an offer with simulcast that lists a number of simulcast versions, MAY reduce the number of simulcast versions in the answer, but MUST NOT add simulcast versions.

An offerer that receives an answer where some simulcast version alternatives are kept MUST be prepared to receive any of the kept send direction alternatives, and MAY send any of the kept receive direction alternatives from the answer. This is similar to the case when the answer includes multiple formats on the m-line.

An offerer that receives an answer where some of the simulcast versions are removed MAY release the corresponding resources (codec, transport, etc) in its receive direction and MUST NOT send any RTP streams corresponding to the removed simulcast versions.

The media formats and corresponding characteristics of encoded streams used in a simulcast SHOULD be chosen such that they are different. If this difference is not required, RTP duplication [RFC7104] procedures SHOULD be considered instead of simulcast.

Note: The inclusion of "a=simulcast" or the use of simulcast does not change any of the interpretation or Offer/Answer procedures for other SDP attributes, like "a=fmtp".

6.2. Relating Simulcast Versions

As long as there is only a single media source per SDP media description, simulcast RTP streams can be related on RTP level through the RTP payload type, as specified in the SDP "a=simulcast" attribute (Section 6.1) parameters. When using BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] to use multiple SDP media descriptions to specify a single RTP session, there is an identification mechanism that allows relating RTP streams back to individual media descriptions, after which the above RTP payload type relation can be used.

6.3. Signaling Examples

These examples are for a case of client to video conference service using a centralized media topology with an RTP mixer.

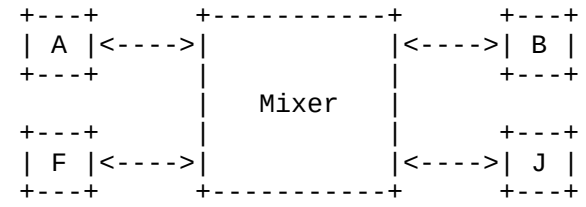


Figure 2: Four-party Mixer-based Conference

6.3.1. Unified Plan Client

Alice is calling in to the mixer with a simulcast-enabled Unified Plan client capable of a single media source per media type. The client can send a simulcast of 2 video resolutions and frame rates: HD 1280x720p 30fps and thumbnail 320x180p 15fps. Alice’s Offer:

```
v=0
o=alice 2362969037 2362969040 IN IP4 192.0.2.156
s=Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.156
m=audio 49200 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 49300 RTP/AVP 97 98
a=rtpmap:97 H264/90000
a=rtpmap:98 H264/90000
a=fmtp:97 profile-level-id=42c01f; max-fs=3600; max-mbps=108000
a=fmtp:98 profile-level-id=42c00b; max-fs=240; max-mbps=3600
a=imageattr:97 send [x=1280,y=720] recv [x=1280,y=720]
a=imageattr:98 send [x=320,y=180] recv [x=320,y=180]
a=simulcast send 97;98 recv 97
```

Figure 3: Unified Plan Simulcast Offer

The only thing in the SDP that indicates simulcast capability is the line in the video media description containing the "simulcast" attribute. The included format parameters indicates that sent simulcast versions can differ in video resolution and framerate.

The Answer from the server indicates that it too is simulcast capable. Should it not have been simulcast capable, the "a=simulcast" line would not have been present and communication would have started with the media negotiated in the SDP.

```
v=0
o=server 823479283 1209384938 IN IP4 192.0.2.2
s=Answer to Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.43
m=audio 49672 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 49674 RTP/AVP 97 98
a=rtpmap:97 H264/90000
a=rtpmap:98 H264/90000
a=fmtp:97 profile-level-id=42c01f; max-fs=3600; max-mbps=108000
a=fmtp:98 profile-level-id=42c00b; max-fs=240; max-mbps=3600
a=imageattr:97 send [x=1280,y=720] recv [x=1280,y=720]
a=imageattr:98 send [x=320,y=180] recv [x=320,y=180]
a=simulcast recv 97;98 send 97
```

Figure 4: Unified Plan Simulcast Answer

Since the server is the simulcast media receiver, it reverses the direction of the "simulcast" attribute.

6.3.2. Multi-Source Client

Fred is calling in to the same conference as in the example above with a two-camera, two-display system, thus capable of handling two separate media sources in each direction, where each media source is simulcast-enabled in the send direction. Fred's client is a Unified Plan client, restricted to a single media source per media description.

The first two simulcast versions for the first media source use different codecs, H264-SVC [RFC6190] and H264 [RFC6184]. These two simulcast versions also have a temporal dependency. Two different video codecs, VP8 [I-D.ietf-payload-vp8] and H264, are offered as alternatives for the third simulcast version for the first media source.

The second media source is offered with three different simulcast versions. All video streams of this second media source are loss protected by RTP retransmission [RFC4588].

Fred's client is also using BUNDLE to send all RTP streams from all media descriptions in the same RTP session on a single media transport. There are not so many RTP payload types in this example that there is any risk of running out of payload types, but for the sake of making an example, it is assumed that one of the payload types cannot be kept unique across all media descriptions. Therefore, the SDP makes use of the mechanism (work in progress) in BUNDLE that identifies which media description an RTP stream belongs to (a new RTCP SDES item and RTP header extension [RFC5285] type carrying the a=mid value). That identification will make it possible to identify unambiguously also on RTP level which media source it is and thus what the related simulcast versions are, even though two separate RTP streams in the joint RTP session share RTP payload type.

```
v=0
o=fred 238947129 823479223 IN IP4 192.0.2.125
s=Offer from Simulcast Enabled Multi-Source Client
t=0 0
c=IN IP4 192.0.2.125
a=group:BUNDLE foo bar zen

m=audio 49200 RTP/AVP 99
a=mid:foo
a=rtpmap:99 G722/8000
```

```
m=video 49600 RTP/AVP 100 101 102 103
a=mid:bar
a=rtpmap:100 H264-SVC/90000
a=rtpmap:101 H264/90000
a=rtpmap:102 H264/90000
a=rtpmap:103 VP8/90000
a=fmtp:100 profile-level-id=42400d; max-fs=3600; max-mbps=108000; \
    mst-mode=NI-TC
a=fmtp:101 profile-level-id=42c00d; max-fs=3600; max-mbps=54000
a=fmtp:102 profile-level-id=42c00d; max-fs=900; max-mbps=27000
a=fmtp:103 max-fs=900; max-fr=30
a=imageattr:100 send [x=1280,y=720] recv [x=1280,y=720]
a=imageattr:101 send [x=1280,y=720] recv [x=1280,y=720]
a=imageattr:102 send [x=640,y=360] recv [x=640,y=360]
a=imageattr:103 send [x=640,y=360] recv [x=640,y=360]
a=depend:100 lay bar:101
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
a=simulcast sendrecv 100;101 send 103,102

m=video 49602 RTP/AVP 96 103 97 104 105 106
a=mid:zen
a=rtpmap:96 VP8/90000
a=fmtp:96 max-fs=3600; max-fr=30
a=rtpmap:104 rtx/90000
a=fmtp:104 apt=96;rtx-time=200
a=rtpmap:103 VP8/90000
a=fmtp:103 max-fs=900; max-fr=30
a=rtpmap:105 rtx/90000
a=fmtp:105 apt=103;rtx-time=200
a=rtpmap:97 VP8/90000
a=fmtp:97 max-fs=240; max-fr=15
a=rtpmap:106 rtx/90000
a=fmtp:106 apt=97;rtx-time=200
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
a=simulcast send 97;96;103
```

Figure 5: Fred’s Multi-Source Simulcast Offer

Note: Empty lines in the SDP above are added only for readability and would not be present in an actual SDP.

7. Network Aspects

Simulcast is in this memo defined as the act of sending multiple alternative encoded streams of the same underlying media source. When transmitting multiple independent streams that originate from the same source, it could potentially be done in several different

ways using RTP. A general discussion on considerations for use of the different RTP multiplexing alternatives can be found in Guidelines for Multiplexing in RTP [I-D.ietf-avtcore-multiplex-guidelines]. Discussion and clarification on how to handle multiple streams in an RTP session can be found in [I-D.ietf-avtcore-rtp-multi-stream].

The network aspects that are relevant for simulcast are:

Quality of Service: When using simulcast it might be of interest to prioritize a particular simulcast version, rather than applying equal treatment to all versions. For example, lower bit-rate versions may be prioritized over higher bit-rate versions to minimize congestion or packet losses in the low bit-rate versions. Thus, there is a benefit to use a simulcast solution that supports QoS as good as possible. By separating simulcast versions into different RTP sessions and send those RTP sessions over different media transports, a simulcast version can be prioritized by existing flow based QoS mechanisms. When using unicast, QoS mechanisms based on individual packet marking are also feasible, which do not require separation of simulcast versions into different RTP sessions to apply different QoS. The proposed solution can be extended to support this functionality with an optional mid: prefix before the RTP payload types of a simulcast version, to describe simulcast across multiple media descriptions.

NAT/FW Traversal: Using multiple RTP sessions will incur more cost for NAT/FW traversal unless they can re-use the same transport flow, which can be achieved by either one of multiplexing multiple RTP sessions on a single lower layer transport [I-D.westerlund-avtcore-transport-multiplexing] or Multiplexing Negotiation Using SDP Port Numbers [I-D.ietf-mmusic-sdp-bundle-negotiation]. If flow based QoS with any differentiation is desirable, the cost for additional transport flows is likely necessary.

Multicast: Multiple RTP sessions will be required to enable combining simulcast with multicast. Different simulcast versions have to be separated to different multicast groups to allow a multicast receiver to pick the version it wants, rather than receive all of them. In this case, the only reasonable implementation is to use different RTP sessions for each multicast group so that reporting and other RTCP functions operate as intended. The proposed solution can be extended to support this functionality with an optional mid: prefix before the RTP payload types of a simulcast version, to describe simulcast across multiple media descriptions.

8. IANA Considerations

This document requests to register a new attribute, simulcast.

Formal registrations to be written.

9. Security Considerations

The simulcast capability and configuration attributes and parameters are vulnerable to attacks in signaling.

A false inclusion of the "a=simulcast" attribute may result in simultaneous transmission of multiple RTP streams that would otherwise not be generated. The impact is limited by the media description joint bandwidth, shared by all simulcast versions irrespective of their number. There may however be a large number of unwanted RTP streams that will impact the share of the bandwidth allocated for the originally wanted RTP stream.

A hostile removal of the "a=simulcast" attribute will result in simulcast not being used.

Neither of the above will likely have any major consequences and can be mitigated by signaling that is at least integrity and source authenticated to prevent an attacker to change it.

10. Contributors

Morgan Lindqvist and Fredrik Jansson, both from Ericsson, have contributed with important material to the first versions of this document. Robert Hansen, from Cisco, contributed significantly to subsequent versions.

11. Acknowledgements

12. References

12.1. Normative References

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

[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.

[RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC7104] Begen, A., Cai, Y., and H. Ou, "Duplication Grouping Semantics in the Session Description Protocol", RFC 7104, January 2014.

12.2. Informative References

[I-D.ietf-avtcore-multiplex-guidelines]
Westerlund, M., Perkins, C., and H. Alvestrand, "Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams", draft-ietf-avtcore-multiplex-guidelines-03 (work in progress), October 2014.

[I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-06 (work in progress), October 2014.

[I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-05 (work in progress), November 2014.

[I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-04 (work in progress), January 2015.

[I-D.ietf-avtext-rtp-stream-pause]
Akram, A., Even, R., and M. Westerlund, "RTP Stream Pause and Resume", draft-ietf-avtext-rtp-stream-pause-05 (work in progress), October 2014.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-15 (work in progress), January 2015.

[I-D.ietf-payload-vp8]
Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", draft-ietf-payload-vp8-13 (work in progress), October 2014.

[I-D.roach-mmusic-unified-plan]
Roach, A., Uberti, J., and M. Thomson, "A Unified Plan for Using SDP with Large Numbers of Media Flows", draft-roach-mmusic-unified-plan-00 (work in progress), July 2013.

[I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiplexing Multiple RTP Sessions onto a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-07 (work in progress), October 2013.

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.

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

[RFC4733] Schulzrinne, H. and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals", RFC 4733, December 2006.

[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.

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

[RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, May 2011.

[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.

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

Appendix A. Changes From Earlier Versions

NOTE TO RFC EDITOR: Please remove this section prior to publication.

A.1. Modifications Between Individual Version -00 and WG Version -00

- Added this appendix.

Authors' Addresses

Bo Burman
Ericsson
Kistavagen 25
SE-164 80 Stockholm
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Stockholm
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com

Mo Zanaty
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: mzanaty@cisco.com

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: August 13, 2015

P. Martinsen
T. Reddy
P. Patil
Cisco

February 9, 2015

ICE IPv4/IPv6 Dual Stack Fairness
draft-martinsen-mmusic-ice-dualstack-fairness-02

Abstract

This document provides guidelines on how to make Interactive Connectivity Establishment (ICE) conclude faster in multihomed and IPv4/IPv6 dual-stack scenarios where broken paths exist. The provided guidelines are backwards compatible with the original ICE specification.

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 13, 2015.

Table of Contents

1. Introduction

Applications should take special care to deprioritize network interfaces known to provide unreliable connectivity. For example certain tunnel services might provide unreliable connectivity. The simple guidelines presented here describes how to deprioritize interfaces known by the application to provide unreliable connectivity. This application knowledge can be based on simple metrics like previous connection success/failure rates or a more static model based on interface types like wired, wireless, cellular, virtual, tunnelled and so on.

There is a also need to introduce more fairness in the handling of connectivity checks for different IP address families in dual-stack IPv4/IPv6 ICE scenarios. Section 4.1.2.1 of ICE [RFC5245] points to [RFC3484] for prioritizing among the different IP families. [RFC3484] is obsoleted by [RFC6724] but following the recommendations from the updated RFC will lead to prioritization of IPv6 over IPv4 for the same candidate type. Due to this, connectivity checks for candidates of the same type (host, reflexive or relay) are sent such that an IP address family is completely depleted before checks from the other address family are started. This results in user noticeable setup delays if the path for the prioritized address family is broken.

To avoid such user noticeable delays when either IPv6 or IPv4 path is broken or excessive slow, this specification encourages intermingling the different address families when connectivity checks are performed. Introducing IP address family fairness into ICE connectivity checks will lead to more sustained dual-stack IPv4/IPv6 deployment as users will no longer have an incentive to disable IPv6.

The cost is a small penalty to the address type that otherwise would have been prioritized.

The guidelines outlined in this specification are backward compatible with a standard ICE implementation. This specification only alters the values used to create the resulting checklists in such a way that the core mechanisms from ICE [RFC5245] are still in effect. The introduced fairness might be better, but not worse than what exists today.

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 document uses terminology defined in [RFC5245].

3. Improving ICE Multihomed Fairness

A multihomed ICE agent can potentially send and receive connectivity checks on all available interfaces. To avoid unnecessary delay when performing connectivity checks it would be beneficial to prioritize interfaces known by the agent to provide connectivity.

Candidates from a interface known to the application to provide unreliable connectivity SHOULD get a low candidate priority. This ensures they appear near the end of the candidate list, and would be the last to be tested during the connectivity check phase. This allows candidate pairs more likely to succeed to be tested first.

If the application is unable to get any interface information regarding type or unable to store any relevant metrics, it SHOULD treat all interfaces as if they have reliable connectivity. This ensures all interfaces gets their fair chance to perform their connectivity checks.

4. Improving ICE Dual Stack Fairness

Candidates SHOULD be prioritized such that a long sequence of candidates belonging to the same address family will be intermingled with candidates from an alternate IP family. For example, promoting IPv4 candidates in the presence of many IPv6 candidates such that an IPv4 address candidate is always present after a small sequence of IPv6 candidates, i.e., reordering candidates such that both IPv6 and IPv4 candidates get a fair chance during the connectivity check phase. This makes ICE connectivity checks more responsive to broken path failures of an address family.

An ICE agent can choose an algorithm or a technique of its choice to ensure that the resulting check lists have a fair intermingled mix of IPv4 and IPv6 address families. However, modifying the check list directly can lead to uncoordinated local and remote check lists that result in ICE taking longer to complete or in the worst case scenario fail. The best approach is to modify the formula for calculating the candidate priority value described in ICE [RFC5245] section 4.1.2.1.

Implementations SHOULD prioritize IPv6 candidates by putting some of them first in the the intermingled checklist. This increases the chance of a IPv6 connectivity checks to complete first and be ready for nomination or usage. This enables implementations to follow the intent of [RFC6555]Happy Eyeballs: Success with Dual-Stack Hosts.

5. Compatibility

ICE [RFC5245] section 4.1.2 states that the formula in section 4.1.2.1 SHOULD be used to calculate the candidate priority. The formula is as follows:

```
priority = (2^24)*(type preference) +
           (2^8)*(local preference) +
           (2^0)*(256 - component ID)
```

ICE [RFC5245] section 4.1.2.2 has guidelines for how the type preference and local preference value should be chosen. Instead of having a static local preference value for IPv4 and IPv6 addresses, it is possible to choose this value dynamically in such a way that IPv4 and IPv6 address candidate priorities ends up intermingled within the same candidate type.

It is also possible to dynamically change the type preference in such a way that IPv4 and IPv6 address candidates end up intermingled regardless of candidate type. This is useful if there are a lot of IPv6 host candidates effectively blocking connectivity checks for IPv4 server reflexive candidates.

The list below shows a sorted local candidate list where the priority is calculated in such a way that the IPv4 and IPv6 candidates are intermingled. To allow for earlier connectivity checks for the IPv4 server reflexive candidates, some of the IPv6 host candidates are demoted. This is just an example of how a candidate priorities can be calculated to provide better fairness between IPv4 and IPv6 candidates without breaking any of the ICE connectivity checks.

	Candidate Type	Address Type	Component ID	Priority
(1)	HOST	IPv6	(1)	2129289471
(2)	HOST	IPv6	(2)	2129289470
(3)	HOST	IPv4	(1)	2129033471
(4)	HOST	IPv4	(2)	2129033470
(5)	HOST	IPv6	(1)	2128777471
(6)	HOST	IPv6	(2)	2128777470
(7)	HOST	IPv4	(1)	2128521471
(8)	HOST	IPv4	(2)	2128521470
(9)	HOST	IPv6	(1)	2127753471
(10)	HOST	IPv6	(2)	2127753470
(11)	SRFLX	IPv6	(1)	1693081855
(12)	SRFLX	IPv6	(2)	1693081854
(13)	SRFLX	IPv4	(1)	1692825855
(14)	SRFLX	IPv4	(2)	1692825854
(15)	HOST	IPv6	(1)	1692057855
(16)	HOST	IPv6	(2)	1692057854
(17)	RELAY	IPv6	(1)	15360255
(18)	RELAY	IPv6	(2)	15360254
(19)	RELAY	IPv4	(1)	15104255
(20)	RELAY	IPv4	(2)	15104254

SRFLX = server reflexive

Note that the list does not alter the component ID part of the formula. This keeps the different components (RTP and RTCP) close in the list. What matters is the ordering of the candidates with component ID 1. Once the checklist is formed for a media stream the candidate pair with component ID 1 will be tested first. If ICE connectivity check is successful then other candidate pairs with the same foundation will be unfrozen ([RFC5245] section 5.7.4. Computing States).

The local and remote agent can have different algorithms for choosing the local preference and type preference values without impacting the synchronization between the local and remote check lists.

The check list is made up by candidate pairs. A candidate pair is two candidates paired up and given a candidate pair priority as described in [RFC5245] section 5.7.2. Using the pair priority formula:

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

Where G is the candidate priority provided by the controlling agent and D the candidate priority provided by the controlled agent. This ensures that the local and remote check lists are coordinated.

Even if the two agents have different algorithms for choosing the candidate priority value to get an intermingled set of IPv4 and IPv6 candidates, the resulting checklist, that is a list sorted by the pair priority value, will be identical on the two agents.

The agent that has promoted IPv4 cautiously i.e. lower IPv4 candidate priority values compared to the other agent, will influence the check list the most due to (2^32*MIN(G,D)) in the formula.

These recommendations are backward compatible with a standard ICE implementation. The resulting local and remote checklist will still be synchronized. The introduced fairness might be better, but not worse than what exists today

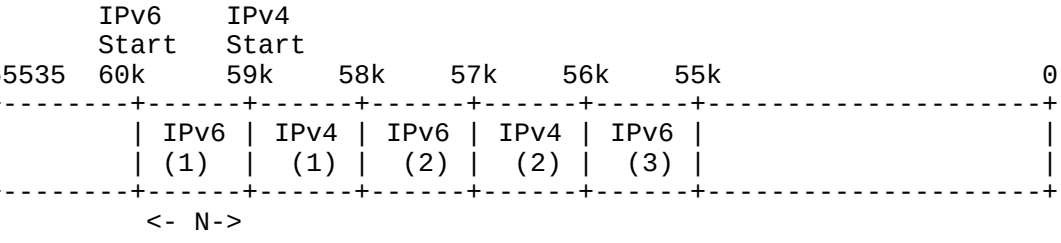
6. Example Algorithm for Choosing the Local Preference

The algorithm described in this section can be used by an implementation to introduce IPv4/IPv6 dual stack and multihomed fairness. Implementations implementing their own algorithm must take care not to break any ICE compatibility. See Section Section 5 for details.

The value space for the local preference is from 0 to 65535 inclusive. This value space can be divided up in chunks for each IP address family.

An IPv6 and IPv4 start priority must be given. In this example IPv6 starts at 60000 and IPv4 at 59000. IPv6 should be given the highest start priority.

Interfaces known to the application to provide unreliable connectivity will be given a low local_preference value. This will place candidates from those interface near the end in a sorted candidate list.



The local preference can be calculated by the given formula:

local_preference = ((S - N*2*(Cn/Cmax))* Ri) + I

S: Address type specific start value (IPv4 or IPv6 Start)

N: Absolute value of IPv6_start-IPv4_start. This ensures a positive number even if IPv4 is the highest priority.

Cn: Number of current candidates of a specific IP address type and candidate type (host, server reflexive or relay).

Cmax: Number of allowed consecutive candidates of the same IP address type.

Ri: Reliable interface. A reliable interface known by the application to provide reliable connectivity should set this value to 1. Interfaces known to provide unreliable connectivity should set this to 0. (Allowed values are 0 and 1)

I: Interface priority. Unreliable interfaces can set this value to get a priority among the unreliable interfaces. Max value is recommended to be N. Reliable interfaces should set this to 0.

Using the values N=abs(60000-59000) and Cmax = 2 yields the following sorted local candidate list with only reliable interfaces:

(1) HOST IPv6 (1) Priority: 2129289471
(2) HOST IPv6 (2) Priority: 2129289470
(3) HOST IPv4 (1) Priority: 2129033471
(4) HOST IPv4 (2) Priority: 2129033470
(5) HOST IPv6 (1) Priority: 2128777471
(6) HOST IPv6 (2) Priority: 2128777470
(7) HOST IPv4 (1) Priority: 2128521471
(8) HOST IPv4 (2) Priority: 2128521470
(9) HOST IPv6 (1) Priority: 2128265471
(10) HOST IPv6 (2) Priority: 2128265470
(11) SRFLX IPv6 (1) Priority: 1693081855
(12) SRFLX IPv6 (2) Priority: 1693081854
(13) SRFLX IPv4 (1) Priority: 1692825855
(14) SRFLX IPv4 (2) Priority: 1692825854
(15) RELAY IPv6 (1) Priority: 15360255
(16) RELAY IPv6 (2) Priority: 15360254
(17) RELAY IPv4 (1) Priority: 15104255
(18) RELAY IPv4 (2) Priority: 15104254

The result is an even spread of IPv6 and IPv4 candidates among the different candidate types (host, server reflexive, relay). The local preference value is calculated separately for each candidate type.

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. Acknowledgements

Authors would like to thank Dan Wing, Ari Keranen, Bernard Aboba, Martin Thomson, Jonathan Lennox, Balint Menyhart and Simon Perreault for their comments and review.

10. Normative References

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

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

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

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

Authors' Addresses

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens Vei 22
Lysaker, Akershus 1325
Norway

Email: palmarti@cisco.com

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.
Bangalore
India

Email: praspati@cisco.com

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: August 5, 2015

S. Nandakumar
Cisco Systems Inc

February 1, 2015

IANA registrations of SDP 'proto' attribute for transporting RTP Media over TCP under various RTP profiles. draft-nandakumar-mmusic-proto-iana-registration-01

Abstract

RTP provides end-to-end network transport functions suitable for applications transmitting real-time data such as audio, video or simulation data, over multicast or unicast network services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality.

The RTP specification [RFC3550] establishes a registry of profile names for use by higher-level control protocols, such as the SDP, to refer to the transport methods. This specification describes the following new SDP transport protocol identifiers for transporting RTP Media over TCP: 'TCP/RTP/AVPF', 'TCP/RTP/SAVP', 'TCP/RTP/SAVPF', 'TCP/DTLS/RTP/SAVP', 'TCP/DTLS/RTP/SAVPF', 'TCP/TLS/RTP/AVP', 'TCP/TLS/RTP/AVPF', 'TCP/TLS/RTP/SAVP', 'TCP/TLS/RTP/SAVPF'.

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 5, 2015.

Table of Contents

1. Overview

SDP [RFC4566] provides a general-purpose format for describing multimedia sessions in announcements or invitations. [RFC4145] specifies a general mechanism for describing media transport over TCP using SDP with [RFC4571] defining a method for framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) packets onto a connection-oriented transport (such as TCP) . [RFC4572] extends [RFC4145] for describing TCP-based media streams that are protected using TLS [RFC5246].

This specification describes the following new SDP transport protocol identifiers for transporting RTP Media over TCP:

TCP/RTP/AVPF: to describe RTP Media with RTCP-based Feedback [RFC4585] over TCP, as defined in Section 3.1.

TCP/RTP/SAVP: to describe Secure RTP Media [RFC3711] over TCP, as defined in Section 3.2.

TCP/RTP/SAVPF: to describe Secure RTP Media with RTCP-based Feedback [RFC5124] over TCP, as defined in Section 3.3.

TCP/DTLS/RTP/SAVP: to describe Secure RTP Media [RFC3711] using DTLS-SRTP [RFC5764] over TCP, as defined in Section 3.4.

TCP/DTLS/RTP/SAVPF: to describe Secure RTP Media with RTCP-based Feedback [RFC5124] using DTLS-SRTP over TCP, as defined in Section 3.5.

TCP/TLS/RTP/AVP: to describe RTP Media on top of TLS over TCP, as defined in Section 3.6.

TCP/TLS/RTP/AVPF: to describe RTP Media with RTCP-based Feedback [RFC5124] on top of TLS over TCP, as defined in Section 3.7.

TCP/TLS/RTP/SAVP: to describe Secure RTP Media on top of TLS over TCP, as defined in Section 3.8.

TCP/TLS/RTP/SAVPF: to describe Secure RTP Media with RTCP-based Feedback [RFC5124] on top of TLS over TCP, as defined in Section 3.9.

2. Terminology

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

3. Protocol Identifiers

The ’m=’ line in SDP specifies, among other items, the transport protocol to be used for the media in the session. See the "Media Descriptions" section of SDP [RFC4566] for a discussion on transport protocol identifiers.

The following is the format for an ’m=’ line, as specified in [RFC4566]:

m=<media> <port> <proto> <fmt> ...

An ’m’ line that specifies these new proto identifiers MUST further qualify the application-layer protocol using an fmt identifier.

3.1. TCP/RTP/AVPF Transport Realization

The TCP/RTP/AVPF is realized as described below:

- RTP/AVPF stream over the TCP transport is realized using the framing method defined in [RFC4571].

3.2. TCP/RTP/SAVP Transport Realization

The TCP/RTP/SAVP is realized as described below:

- RTP/SAVP stream over the TCP transport is realized using the framing method defined in [RFC4571].

3.3. TCP/RTP/SAVPF Transport Realization

The TCP/RTP/SAVPF is realized as described below:

- RTP/SAVPF stream over the TCP transport is realized using the framing method defined in [RFC4571].

3.4. TCP/DTLS/RTP/SAVP Transport Realization

The TCP/DTLS/RTP/SAVP is realized as described below:

- RTP/SAVP on top of DTLS is realized according to the procedures defined in [RFC5764]; and

- o [RFC4571] framing is used to transport DTLS-SRTP packets over TCP.

3.5. TCP/DTLS/RTP/SAVPF Transport Realization

The TCP/DTLS/RTP/SAVPF is realized as described below:

- o RTP/SAVPF on top of DTLS is realized according to the procedures defined in [RFC5764]; and

- o [RFC4571] framing is used to transport DTLS-SRTP packets over TCP.

3.6. TCP/TLS/RTP/AVP Transport Realization

The TCP/TLS/RTP/AVP is realized as described below:

- o RTP/AVP packets are framed using the procedures from [RFC4571]; and

- o [RFC4571] framed RTP/AVP packets are transported as Application data messages over the TLS association setup using the procedures from [RFC4572].

3.7. TCP/TLS/RTP/AVPF Transport Realization

The TCP/TLS/RTP/AVPF is realized as described below:

- o RTP/AVPF packets are framed using the procedures from [RFC4571]; and

- o [RFC4571] framed RTP/AVPF packets are transported as Application data messages over the TLS association setup using the procedures from [RFC4572].

3.8. TCP/TLS/RTP/SAVP Transport Realization

The TCP/TLS/RTP/SAVP is realized as described below:

- o [RFC4572] procedures are followed for setting up TLS association(s) between the peers. However, the cryptographic mechanism used to generate the certificate fingerprint presented in the SDP MUST be chosen from the SRTPProtectionProfiles as described in [RFC5764]; and

- o RTP/SAVP packets are framed according to the procedures from [RFC4571]; and

- o [RFC4571] framed RTP/SAVP packets are transported as Application data messages over the TLS association setup using the procedures from [RFC4572].

3.9. TCP/TLS/RTP/SAVPF Transport Realization

The TCP/TLS/RTP/SAVPF is realized as described below:

- o [RFC4572] procedures are followed for setting up TLS association(s) between the peers. However, the cryptographic mechanism used to generate the certificate fingerprint presented in the SDP MUST be chosen from the SRTPProtectionProfiles as described in [RFC5764]; and

- o RTP/SAVPF packets are framed according to the procedures from [RFC4571]; and

- o [RFC4571] framed RTP/SAVPF packets are transported as Application data messages over the TLS association setup using the procedures from [RFC4572].

4. ICE Considerations

When procedures from [RFC6544] are used to setup ICE [RFC5245] candidates for a TCP transport, the framing mechanism from [RFC4571] is used for STUN keep-alive packets as well, as defined in section 3 of [RFC6544].

5. IANA Considerations

This specification describes the following new SDP transport protocol identifiers :'TCP/RTP/AVPF', 'TCP/RTP/SAVP', 'TCP/RTP/SAVPF', 'TCP/DTLS/RTP/SAVP', 'TCP/DTLS/RTP/SAVPF', 'TCP/TLS/RTP/AVP', 'TCP/TLS/RTP/AVPF', 'TCP/TLS/RTP/SAVP', 'TCP/TLS/RTP/SAVPF' as defined in the Section 3. These proto values should be registered by the IANA under the:

- o "proto" subregistery in the "Session Description Protocol (SDP) Parameters" registry; and

- o "RTP Profile Names" registry subregistry on the "Real-Time Transport Protocol (RTP) Parameters" registry.

Additionally the following proto values described in [RFC5764] should be registered under the "RTP Profile Names" subregistry under the "Real-Time Transport Protocol (RTP) Parameters" registry: 'UDP/TLS/RTP/SAVP' , 'DCCP/TLS/RTP/SAVP', 'UDP/TLS/RTP/SAVPF', 'DCCP/TLS/RTP/SAVPF'.

6. Security Considerations

The new "proto" identifiers registered by this document in the SDP parameters registry maintained by IANA is primarily for use by the offer/answer model of the Session Description Protocol [RFC3264] for the negotiation and establishment of RTP based Media over the TCP transport. These additional SDP "proto" identifiers does not introduce any security considerations beyond those detailed in Section 7 of [RFC4566].

7. Acknowledgements

Author would like to thank Cullen Jennings, Alissa Cooper, Justin Uberti and Christer Holmberg for early reviews and suggested improvements.

8. References

8.1. Normative References

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

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

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, 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.

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

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

8.2. Informative References

[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.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

[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.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Author’s Address

Suhas Nandakumar
Cisco Systems Inc
707 Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

J. Uberti
Google
J. Lennox
Vidyo

March 09, 2015

Improvements to ICE Candidate Nomination
draft-uberti-mmusic-nombis-00

Abstract

This document makes recommendations for simplifying and improving the procedures for candidate nomination in Interactive Connectivity Establishment (ICE).

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Table of Contents

1. Introduction

Interactive Connectivity Establishment (ICE) attempts to find the 'best' path for connectivity between two peers; in ICE parlance, these paths are known as 'candidate pairs'. During the ICE process, one endpoint, known as the 'controlling' endpoint, selects a candidate pair as the best pair; this action is known as nomination. ICE supports two different mechanisms for performing nomination, known as Regular Nomination, and Aggressive Nomination.

However, each of these modes have flaws that restrict their usefulness. Regular Nomination, as currently speced, requires a best pair to be chosen before media transmission can start, causing unnecessary call setup delay. Aggressive Nomination, while avoiding this delay, gives the controlling endpoint much less discretion into

which candidate pair is chosen, preventing it from making decisions based on dynamic factors such as RTT or loss rate. Needless to say, the presence of both modes also adds nontrivial complexity.

Lastly, ICE is currently defined as a finite process, where the decision on the optimal candidate pair is made during call setup and infrequently (if ever) changed. While this may be acceptable for endpoints with static network configurations, it fails to meet the needs of mobile endpoints, who may need to seamlessly move between networks, or be connected to multiple networks simultaneously. In these cases, the controlling endpoint may want to maintain multiple potential candidate pairs, and make dynamic decisions to switch between them as conditions change.

To address these challenges, this document makes two proposals for refactoring ICE nomination - merging Regular and Aggressive Nomination, and introducing a new mode, known as Continuous Nomination. This makes ICE substantially more flexible without increasing complexity.

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. Goals and Requirements

The goals for improved ICE nomination are enumerated below.

3.1. Minimize Call Setup Latency

Modern ICE agents will often have multiple network interfaces and multiple servers from which to obtain ICE candidates. While some ICE checks may succeed quickly, finishing the entire set of checks can easily take multiple seconds; this concern is discussed in [RFC5245], Section 8.1.1.1. As a result, ICE endpoints MUST be able to start transmitting media immediately upon a successful ICE check, and MUST retain the ability to switch if a better candidate pair becomes available later.

3.2. Allow Controlling Endpoint to Make Dynamic Decisions

While an ICE endpoint will assign various priority values to its ICE candidates, these priorities are static and can only be based on a priori knowledge; the shortcomings of this approach are discussed in the first paragraph of Section 2.6 in [RFC5245]. To properly make choices in multi-network and multi-server scenarios, the controlling

endpoint MUST be able to make dynamic decisions about the selected candidate pair based on observed network performance. For example, RTT could be used to evaluate which TURN servers to use, as described in [I-D.williams-peer-redirect] To ensure symmetric flows, this implies that the controlling endpoint MUST be able to communicate its choice to the controlled side.

3.3. Allow Selected Pair Change At Any Time Without Signaling

Expanding on the requirement above, the need to make dynamic decisions is not limited to call setup. A multihomed endpoint may need to switch interfaces based on mobility considerations, or a robust endpoint may want to keep multiple network paths warm and switch immediately if connectivity is interrupted on one of them. As the signaling channel may be affected by the event necessitating the switch, this implies that the controlling endpoint MUST be able to change the selected pair and indicate this to the remote side without signaling. The need for this functionality has been stated in [I-D.wing-mmusic-ice-mobility] and [I-D.singh-avtcore-mprtp].

The rules in [RFC5245] ensure that the controlled endpoint keeps its candidate needed for the selected pair alive. However, in order for alternate pairs to remain available, the controlled endpoint must keep the associated candidates alive as well, following the procedures outlined in [RFC5245], Section 4.1.1.4. This implies that the controlling endpoint MUST have some way to indicate to the controlled side that specific candidates are to be kept alive.

3.4. Allow Continuous Addition of Candidates

In certain network mobility scenarios, networks may come up and down while the call is active. In order to allow candidates gathered on newly available networks to be used for the selected pair or backup pairs, the endpoint MUST be able to gather candidates on these networks and communicate them to the remote side. While this could be done using an ICE restart, as described in [RFC5245], Section 9.1, the ICE restart may have unintended consequences, such as causing the remote side to regather all candidates. Instead, it would be best if the new candidates could be trickled, as discussed in [I-D.ietf-mmusic-trickle-ice], but even after ICE processing has completed.

3.5. Maintain Backwards Compatibility

To prevent interoperability problems, ICE endpoints that support the functionality listed above MUST still maintain [RFC5245] compliance when interacting with existing endpoints. However, the ideal

solution SHOULD allow some improvements to occur when only the controlling side supports the new functionality.

3.6. Minimize Complexity Increase

Increased functionality typically leads to increased complexity, which leads to more edge cases, and more implementation bugs. This suggests that in addition to proposing new ICE functionality, the ideal solution SHOULD deprecate superfluous functionality.

4. Deprecating Aggressive Nomination

4.1. Overview

The main benefits of Regular Nomination are that the controlling side can dynamically choose which candidate pair to use, and a clear signal when the nomination process has completed, via the presence of the USE-CANDIDATE flag in a Binding Request. The main benefit of Aggressive Nomination is that it is only necessary to send a single Binding Request before starting the transmission of media, reducing setup latency. Why don’t we have both?

By preserving the dynamic behavior of Regular Nomination, but allowing media transmission to start upon a single successful connectivity check, as in Aggressive Nomination, the requirements of Section 3.1 and Section 3.2 can be met, while meeting the compatibility requirement from Section 3.5 and, since Aggressive Nomination is no longer needed, the complexity requirement from Section 3.6.

4.2. Operation

Since media may be transmitted as soon as all components have a valid pair, as indicated in [RFC5245], Page 69, an ICE Agent can begin transmitting media as soon as this occurs, even if it has not sent a Binding Request with USE-CANDIDATE.

This pair can change as more pairs are added to the Valid list on the controlling side. When nomination completes, and a final pair is selected, this is communicated to the controlled side via the typical Binding Request with USE-CANDIDATE.

On the controlled side, the same process can occur, with the ICE Agent transmitting media as soon as a valid pair exists. To encourage use of symmetric RTP, the controlled ICE Agent SHOULD use the same candidate pair on which it received media from the controlling side. [Doesn’t need to be secure media, since the

controlling side will finalize this preference through USE-CANDIDATE shortly.]

As this is legal ICE behavior, no negotiation of this mechanism should be needed. In the event the receiver drops any packets that arrive before a Binding Request with USE-CANDIDATE set, this will simply lead to brief media clipping and will resolve itself once nomination completes.

4.3. Backwards Compatibility

When acting in the controlled role, new implementations MUST NOT use Aggressive Nomination.

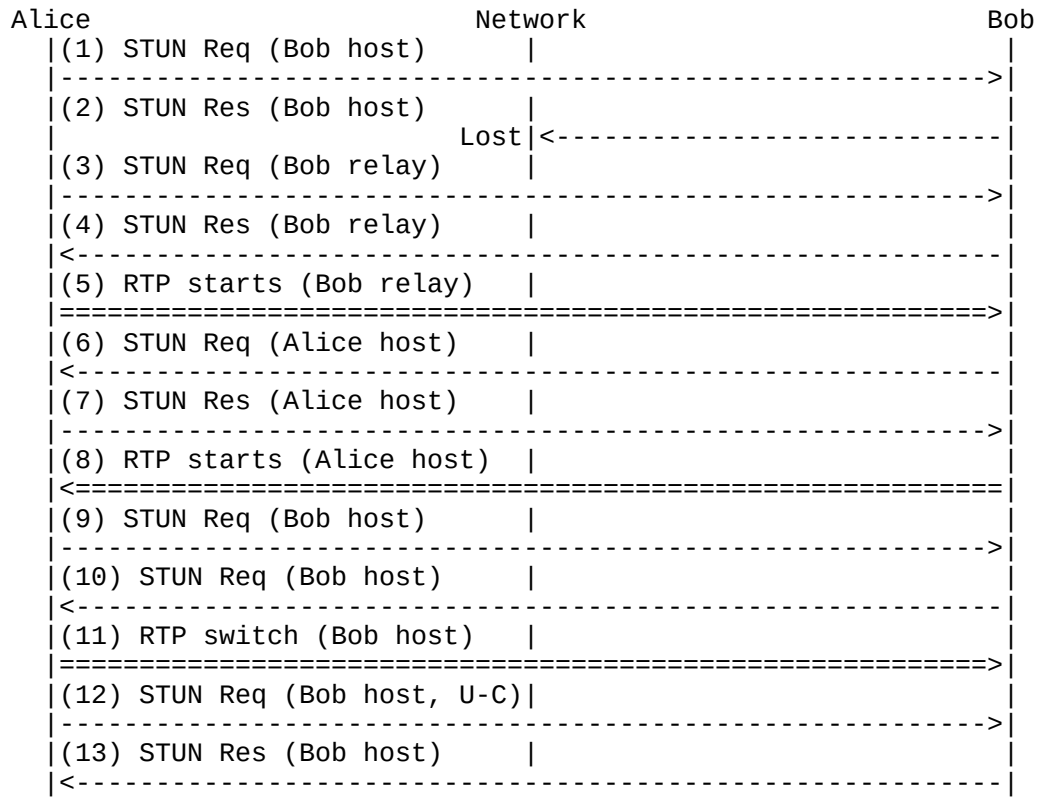
When acting in the controlled role, and the controlling side is using Aggressive Nomination (e.g. sending USE-CANDIDATE in its initial Binding Requests), the standard PRIORITY-based mechanism outlined in [RFC5245], Section 8.1.1.2 should be used to determine the reverse media path.

Note that if implementations would prefer to just avoid Aggressive Nomination altogether, they MAY indicate some TBD pseudo-option in the ice-options attribute. Because compliant implementations MUST NOT use Aggressive Nomination if an unknown ICE option is encountered, this effectively prohibits the use of Aggressive Nomination. [N.B. this could be the ice-options:continuous option described below]

4.4. Examples

An example call setup using Regular Nomination as described above is shown here. Alice is in the controlling role, and Bob is in the controlled role; Alice has a single host candidate and Bob has both host and relay candidates.

Alice’s initial check to Bob’s host candidate fails, but the check to his relay candidate succeeds, so Alice starts transmitting media on her host-relay pair. Bob’s initial check from his host candidate to Alice’s host candidate succeeds, so he starts transmitting media over this host-host pair to Alice. However, when Alice’s host check is later retransmitted, it succeeds, and Alice determines that the host-host pair has a better RTT than the host-relay pair, so she cuts media over to use the host-host pair. Eventually, Alice concludes Regular Nomination by sending a final check to Bob with the USE-CANDIDATE flag set. If Bob had selected a different pair to use than Alice, this action would have forced Bob to use the same pair.



5. Introducing Continuous Nomination

5.1. Overview

As discussed above, in mobile environments there can be multiple possible valid candidate pairs, and these can change at various points in the call, as new interfaces go up and down, signal strength for wireless interfaces changes, and new relay servers are discovered.

However, under 5245 rules, once a candidate pair is selected and confirmed, via USE-CANDIDATE, nomination has completed and cannot be restarted without performing an ICE restart. This is overly complex in many cases, and especially problematic in some specific ones, namely a wifi-cellular handover, where the signaling path for communicating an ICE restart may be impacted by the handover.

To address this situation, this section introduces the concept of "continuous nomination", where the controlling ICE endpoint can adjust the selected candidate pair at any time. By allowing ICE

processing to occur continuously during a call, rather than just at call setup, the requirements expressed in Section 3.3 and Section 3.4 can be met.

5.2. Operation

Under continuous nomination, ICE never concludes; new candidates can always be trickled, and a new candidate pair can be selected by the controlling side at any time.

When selecting a new candidate pair, the controlling side informs the controlled side of the chosen path by sending a new Binding Request with a USE-CANDIDATE attribute. The decision about which candidate pair to use is fully dynamic; the controlling side can use metrics such as RTT or loss rate to change the selected pair at any time. If Binding Requests need to be sent for any other reason, such as consent checks [I-D.ietf-rtcweb-stun-consent-freshness], any checks sent on the selected pair MUST include a USE-CANDIDATE attribute.

Upon receipt of a Binding Request with USE-CANDIDATE, the controlled side MUST switch its media path to the candidate pair on which the Binding Request was received.

During continuous nomination, the controlling side may still elect to prune certain candidate pairs; for example, an implementation may choose to drop relay candidates once a successful connection has been established. The controlled side, however, should follow the controlling side’s lead in terms of deciding whether any pairs should be pruned. [TODO: should the controlled side have any say in the matter, e.g. to eliminate certain candidates?] The controlling ICE Agent informs the remote side of its preferences by continuing to send Binding Requests to the remote side on each candidate pair that it wants to retain. The controlled ICE Agent SHOULD prune any candidate pairs that have not received a Binding Request in N seconds (30?), and SHOULD NOT keep alive any candidates that are not associated with a live candidate pair. [TODO: decide if this implicit timeout approach is correct, or if we should have some sort of approach similar to TURN LIFETIME indicating when a pair should be GCed, with LIFETIME==0 indicating immediate GC.] One side benefit of doing this is that the continuous exchange of Binding Requests across all candidate pairs allows the RTT and loss rate for each to be reliably determined and kept up to date.

If the endpoints have negotiated Trickle ICE support [I-D.ietf-mmusic-trickle-ice], and new candidates become available on either side, the endpoint may send these candidates to the remote side using the existing Trickle ICE mechanisms. Once all of the new candidates have been transmitted, the endpoint MUST send an end-of-

candidates messages, which indicates that no more candidates will be sent in the near future.

At any point, either side may perform an ICE restart, which will result in both sides gathering new ICE candidates, starting a new continuous nomination sequence, and upon successful completion, discarding all candidates from the previous nomination sequence.

5.3. Backwards Compatibility

Since standard ICE implementations may not expect the selected pair to change after a USE-CANDIDATE attribute is received, support for continuous nomination is explicitly indicated via a new "continuous" value for ice-options. If the remote side does not support the "continuous" option, the controlling side MUST fall back to Regular Nomination, as specified in [RFC5245], Sectiom 8.1.1.

5.4. Examples

5.4.1. Switching Between Pairs Based on RTT

Alice and Bob have set up a call using ICE and have established multiple valid pairs. The currently selected pair is for a peer-to-peer route, as it had the highest initial priority value. However, they have also kept alive a selected pair that goes through their TURN servers. At a certain point, Alice detects, via the connectivity checks that she continues to do on the relayed pair, that it actually has a better RTT than the peer-to-peer path. She then decides to switch media over to this path.

As mentioned above, this is easily handled by Alice immediately switching her media to the relayed path; future STUN checks on this path also include the USE-CANDIDATE attribute.

5.4.2. Switching To A New TURN Server

Alice and Bob have set up a call using ICE, and are currently sending their media through Alice’s TURN server. At a certain point, Alice’s application discovers a new TURN server that it thinks might provide a better path for this call.

Alice gathers new candidates from this TURN server, and trickles them to Bob. They perform connectivity checks using these candidates, and Alice determines that the RTT when going through this TURN server is better than the RTT of the current relayed path.

As in the previous example, this is easily handled by Alice switching media to the new path, along with sending USE-CANDIDATE. If the old

path is no longer needed, Alice can destroy the allocation on the old TURN server, and Bob will stop checking it when it stops working.

5.4.3. Switching From WLAN to WWAN

Alice and Bob have set up a call using ICE, and are currently exchanging their media directly via a peer-to-peer path. Alice is on a mobile device, with both wifi and cellular interfaces, but for power reasons, candidates have only been gathered on the wifi interface. At a certain point, Alice leaves her home while the call is active.

In response to the decreasing wifi signal strength, Alice starts to collect candidates on the cellular interface, and trickles them to Bob. They perform connectivity checks using these candidates, and, because of the low wifi signal strength, these candidates are preferred over the existing selected pair.

As in the previous examples, Alice can easily switch media to the new selected pair. When Alice walks completely out of wifi range, and the wifi interface goes down, the wifi candidates are pruned, and any valid pairs on Bob’s side that use those candidates will time out and be pruned as well.

6. Security Considerations

TODO

7. IANA Considerations

A new ICE option "continuous" has been [will be] registered in the "ICE Options" registry created by [RFC6336].

8. Acknowledgements

Several people provided significant input into this document, including Martin Thomson, Brandon Williams, and Dan Wing. Emil Ivov also provided several of the examples for continuous nomination.

9. References

9.1. Normative References

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

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

[RFC6336] Westerlund, M. and C. Perkins, "IANA Registry for Interactive Connectivity Establishment (ICE) Options", RFC 6336, July 2011.

9.2. Informative References

[I-D.ietf-mmusic-trickle-ice]
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", draft-ietf-mmusic-trickle-ice-02 (work in progress), January 2015.

[I-D.ietf-rtcweb-stun-consent-freshness]
Perumal, M., Wing, D., R, R., Reddy, T., and M. Thomson, "STUN Usage for Consent Freshness", draft-ietf-rtcweb-stun-consent-freshness-11 (work in progress), December 2014.

[I-D.singh-avtcore-mprtp]
Singh, V., Karkkainen, T., Ott, J., Ahsan, S., and L. Eggert, "Multipath RTP (MPRTP)", draft-singh-avtcore-mprtp-10 (work in progress), November 2014.

[I-D.williams-peer-redirect]
Williams, B. and T. Reddy, "Peer-specific Redirection for Traversal Using Relays around NAT (TURN)", draft-williams-peer-redirect-03 (work in progress), December 2014.

[I-D.wing-mmusic-ice-mobility]
Wing, D., Reddy, T., Patil, P., and P. Martinsen, "Mobility with ICE (MICE)", draft-wing-mmusic-ice-mobility-07 (work in progress), June 2014.

Appendix A. Change log

Changes in draft -00:

- Initial version, from mailing list discussion post-IETF 90.

Authors' Addresses

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

Email: justin@uberti.name

Jonathan Lennox
Vidyo
433 Hackensack Avenue
Hackensack, NJ 07601
USA

Email: jonathan@vidyo.com