

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 8, 2012

J. Goldberg
Cisco
M. Westerlund
Ericsson
T. Zeng
Nextwave Wireless, Inc.
May 7, 2012

**A Network Address Translator (NAT) Traversal mechanism for media
controlled by Real-Time Streaming Protocol (RTSP)
draft-ietf-mmusic-rtsp-nat-12**

Abstract

This document defines a solution for Network Address Translation (NAT) traversal for datagram based media streams setup and controlled with Real-time Streaming Protocol version 2 (RTSP 2.0). It uses Interactive Connectivity Establishment (ICE) adapted to use RTSP as a signalling channel, defining the necessary extra RTSP extensions and procedures.

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 November 8, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1.	Introduction	4
2.	Definitions	4
3.	Solution Overview	5
4.	RTSP Extensions	7
4.1.	ICE Transport Lower Layer	7
4.2.	ICE Candidate Transport Header Parameter	8
4.3.	ICE Password and Username Transport Header Parameters	11
4.4.	ICE Feature Tag	11
4.5.	Status Codes	12
4.5.1.	150 ICE connectivity checks in progress	12
4.5.2.	480 ICE Processing Failed	12
4.6.	New Reason for PLAY_NOTIFY	12
4.7.	Server Side SDP Attribute for ICE Support	13
4.8.	ICE Features Not Required in RTSP	13
4.8.1.	ICE-Lite	13
4.8.2.	ICE-Mismatch	13
4.8.3.	ICE Remote Candidate Transport Header Parameter	13
5.	Detailed Solution	14
5.1.	Session description and RTSP DESCRIBE (optional)	14
5.2.	Setting up the Media Streams	15
5.3.	RTSP SETUP Request	15
5.4.	Gathering Candidates	16
5.5.	RTSP Server Response	17
5.6.	Server to Client ICE Connectivity Checks	18
5.7.	Client to Server ICE Connectivity Check	18
5.8.	Client Connectivity Checks Complete	18
5.9.	Server Connectivity Checks Complete	19
5.10.	Releasing Candidates	19
5.11.	Steady State	19
5.12.	Re-SETUP	19
5.13.	Server Side Changes After Steady State	20
6.	ICE and Proxies	22
6.1.	Media Handling Proxies	22
6.2.	Signalling Only Proxies	22
6.3.	Non-supporting Proxies	23
7.	RTP and RTCP Multiplexing	23
8.	Fallback	24
9.	IANA Considerations	25
9.1.	RTSP Feature Tags	25

9.2.	Transport Protocol Specifications	26
9.3.	RTSP Transport Parameters	26
9.4.	RTSP Status Codes	26
9.5.	Notify-Reason value	27
9.6.	SDP Attribute	27
10.	Security Considerations	27
10.1.	ICE and RTSP	27
11.	Acknowledgements	28
12.	References	28
12.1.	Normative References	28
12.2.	Informative References	29
	Authors' Addresses	30

1. Introduction

Real-time Streaming Protocol (RTSP) [[RFC2326](#)] and RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)] is protocols used to setup and control one or more media streams delivering media to receivers. It is RTSP's functionality of setting up media streams that cause serious issues with Network Address Translators (NAT) [[RFC3022](#)] unless extra provisions are taken by the protocol. There is thus a need for a NAT traversal mechanism for the media setup using RTSP.

RTSP 1.0 [[RFC2326](#)] has suffered from the lack of a standardized NAT traversal mechanism for a long time, however due to quality of the RTSP 1.0 specification, the work was difficult to specify in an interoperable fashion. This document is therefore built on the specification of RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)]. RTSP 2.0 is similar to RTSP 1.0 in many respects but significantly for this work, it contains a well defined extension mechanism so allowing a NAT traversal extension to be defined that is backwards compatible with RTSP 2.0 peers not supporting the extension. This extension mechanism was not possible in RTSP 1.0 as it would break RTSP 1.0 syntax so causing compatibility issues.

There have been a number of suggested ways of resolving the NAT-traversal of media for RTSP of which a large number are already used in implementations. The evaluation of these NAT traversal solutions in [[I-D.ietf-mmusic-rtsp-nat-evaluation](#)] has shown that there are many issues to consider, so after extensive evaluation, a mechanism based on Interactive Connectivity Establishment (ICE) were selected. This was mainly two reasons: Firstly the mechanism supports RTSP servers behind NATs and secondly the mechanism mitigates the security threat of using RTSP servers as Distributed Denial of Service (DDoS) attack tools.

This document specifies an ICE based solution that is optimized for media delivery server to client. If in the future extensions are specified for other delivery modes than "PLAY", then the optimizations in regards to when PLAY request are sent needs to be reconsidered.

The NAT problem for RTSP signalling traffic itself is beyond the scope of this document and is left for future study should the need arise, because it is a less prevalent problem than the NAT problem for RTSP media streams.

2. Definitions

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. Solution Overview

This overview assumes that the reader has some familiarity with how ICE [[RFC5245](#)] in the context of "SIP: Session Initiation Protocol" [[RFC3261](#)] and "An Offer/Answer Model with the Session Description Protocol (SDP)" [[RFC3264](#)] works, as it primarily points out how the different ICE steps are accomplished in RTSP.

1. RTSP server should indicate it has support for ICE via an SDP [[RFC4566](#)] attribute ("a=rtsp-ice-d-m") in, for example, the SDP returned in RTSP DESCRIBE message. This allows RTSP clients to only send the new ICE interchanges with servers that support ICE so as to limit the overhead on current non-ICE supporting RTSP servers. If RTSP DESCRIBE is used the normal capability determination mechanism should also be used, i.e. "Supported" header and the defined feature tag. Note: Both mechanisms should be used as there are use cases when either of them are not used.
2. The RTSP client reviews the session description returned, for example by an RTSP DESCRIBE message, to determine what media streams need to be setup. For each of these media streams where the transport protocol supports Session Traversal Utilities for (NAT) (STUN) [[RFC5389](#)] based connectivity checks, the client gathers candidate addresses. See [section 4.1.1](#) in ICE [[RFC5245](#)]. The client also installs the STUN servers on each of the local candidates.
3. The RTSP client sends SETUP requests with both a transport specification with a lower layer indicating ICE and a new RTSP Transport header parameter "candidates" listing the ICE candidates for each media stream.
4. After receiving the list of candidates from a client, the RTSP server gathers its own candidates. If the server has a public IP address, then a single candidate per address family (e.g. IPv4 and IPv6), media stream and media component tuple can be included to reduce the number of combinations and speed up the completion.
5. The server sets up the media and if successful responds to the SETUP request with a 200 OK response. In that response the server selects the transport specification using ICE and includes its candidates in the candidates parameter.

6. The server starts the connectivity checks following the procedures described in [Section 5.7](#) and 5.8 of ICE [[RFC5245](#)]. If the server has a public IP address with a single candidate per media stream, component and address family then one may configure the server to not initiate connectivity checks.
7. The client receives the SETUP response and learns the candidate address to use for the connectivity checks, and then initiates its connectivity check, following the procedures in [Section 6](#) of ICE [[RFC5245](#)].
8. When a connectivity check from the client reaches the server it will result in a triggered check from the server. This is why servers with a public IP address can wait until this triggered check to send out any checks for itself, so saving resources and mitigating the DDoS potential from server connectivity checks.
9. When the client has concluded its connectivity checks, including promoting candidates, and has correspondingly received the server connectivity checks on the promoted candidates for all mandatory components of all media streams, it can issue a PLAY request. If the connectivity checks have not concluded successfully then the client may send a new SETUP request assuming it has any new information or believes the server may be able to do more that can result in successful checks.
10. When the RTSP servers receives a PLAY request it checks to see the connectivity checks has concluded successfully and only then can it play the stream. If there is a problem with the checks then the server sends to the client either a 150 (ICE connectivity checks in progress) response to show that it is still working on the connectivity checks or a 480 (ICE Processing Failed) response to indicate a failure of the checks. If the checks are successful then the server sends a 200 OK response and starts delivering media.

The client and server may release unused candidates when the ICE processing has concluded and a single candidate per component has been promoted and a PLAY response has been received (Client) or sent (Server).

The client shall continue to use STUN to send keep-alive for the used bindings. This is important as often RTSP media sessions only contain media traffic from the server to the client so the bindings in the NAT needs to be refreshed by the client to server traffic provided by the STUN keep-alive.

4. RTSP Extensions

This section defines the necessary RTSP extensions for performing ICE with RTSP. Note that these extensions are based on the SDP attributes in the ICE specification unless expressly indicated.

4.1. ICE Transport Lower Layer

A new lower layer "D-ICE" for transport specifications is defined. This lower layer is datagram clean except that the protocol used must be demultiplexiable with STUN messages (see STUN [[RFC5389](#)]). With datagram clean we mean that it must be capable of describing the length of the datagram, transport that datagram (as a binary chunk of data) and provide it at the receiving side as one single item. This lower layer can be any transport type defined for ICE which does provide datagram transport capabilities. UDP based transport candidates are defined in ICE [[RFC5245](#)] and MUST be supported. It is OPTIONAL to also support TCP based candidates as defined by "TCP Candidates with Interactive Connectivity Establishment (ICE)" [[RFC6544](#)]. The TCP based candidate fulfills the requirements on providing datagram transport and can thus be used in combination with RTP. Additional transport types for candidates may be defined in the future.

This lower layer uses ICE to determine which of the different candidates shall be used and then when the ICE processing has concluded, uses the selected candidate to transport the datagrams over this transport.

This lower layer transport can be combined with all upper layer media transport protocols that are possible to demultiplex with STUN and which use datagrams. This specification defines the following combinations:

- o RTP/AVP/D-ICE
- o RTP/AVPF/D-ICE
- o RTP/SAVP/D-ICE
- o RTP/SAVPF/D-ICE

This list can easily be extended with more transport specifications after having performed the evaluation that they are compatible with D-ICE as lower layer.

The lower-layer "D-ICE" has the following rules for the inclusion of transport parameters:

unicast: As ICE only supports unicast operations, thus it is REQUIRED that one include the unicast indicator parameter, see [section 16.46](#) in RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)].

candidates: The "candidates" parameter SHALL be included as this specify at least one candidate to try to establish a working transport path with.

dest_addr: This parameter SHALL NOT be included as "candidates" is used instead to provide the necessary address information.

ICE-Password: This parameter SHALL be included.

ICE-ufrag: This parameter SHALL be included.

[4.2.](#) ICE Candidate Transport Header Parameter

This section defines a new RTSP transport parameter for carrying ICE candidates related to the transport specification they appear within, which may then be validated with an end-to-end connectivity check using STUN [[RFC5389](#)]. Transport parameters may only occur once in each transport specification. For transport specification using "D-ICE" as lower layer, this parameter needs to be present. The parameter can contain one or more ICE candidates. In the SETUP response there is only a single transport specification, and if that uses the "D-ICE" lower layer this parameter MUST be present and include the server side candidates.


```

trns-parameter = <Defined in Section 20.2.3 of
                  [I-D.ietf-mmusic-rfc2326bis]>
trns-parameter =/ SEMI ice-trn-par
ice-trn-par     = "candidates" EQUAL DQ SWS ice-candidate
                  *(SEMI ice-candidate) SWS DQ
ice-candidate   = foundation SP
                  component-id SP
                  transport SP
                  priority SP
                  connection-address SP
                  port SP
                  cand-type
                  [SP rel-addr]
                  [SP rel-port]
                  [SP tcp-type-ext] ; Mandatory if transport = TCP
                  *(SP extension-att-name SP extension-att-value)

foundation      = <See section 15.1 of \[RFC5245\]>
component-id    = <See section 15.1 of \[RFC5245\]>
transport       = <See section 15.1 of \[RFC5245\]>
transport-extension = <See section 15.1 of \[RFC5245\]>
priority        = <See section 15.1 of \[RFC5245\]>
cand-type       = <See section 15.1 of \[RFC5245\]>
candidate-types = <See section 15.1 of \[RFC5245\]>
rel-addr        = <See section 15.1 of \[RFC5245\]>
rel-port        = <See section 15.1 of \[RFC5245\]>
tcp-type-ext    = <See section 4.5 of \[RFC6544\]>
extension-att-name = <See section 15.1 of \[RFC5245\]>
extension-att-value = <See section 15.1 of \[RFC5245\]>
ice-char        = <See section 15.1 of \[RFC5245\]>
connection-address = <See \[RFC4566\]>
port            = <See \[RFC4566\]>
EQUAL           = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
DQ              = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
SWS             = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
SEMI            = <Defined in [I-D.ietf-mmusic-rfc2326bis]>

```

<connection-address>: is the IP address of the candidate, allowing for IPv4 addresses, IPv6 addresses and Fully qualified domain names (FQDN), taken from ICE [[RFC4566](#)]. The connection address SHOULD be on the same format (explicit IP or FQDN) as in the dest_addr parameter used to express fallbacks. An IP address SHOULD be used, but an FQDN MAY be used in place of an IP address. In that case, when receiving an SETUP request or response containing an FQDN in an candidate parameter, 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 record. If the DNS query returns more

than one IP address, one is chosen, and then used for the remainder of ICE processing which in RTSP is subsequent RTSP SETUPS for the same RTSP session.

<port>: is the port of the candidate, the syntax defined by SDP [[RFC4566](#)].

<transport>: indicates the transport protocol for the candidate. The ICE specification defines UDP. "TCP Candidates with Interactive Connectivity Establishment (ICE)" [[RFC6544](#)] defines how TCP is used as candidates. Additional extensibility is provided to allow for future transport protocols to be used with ICE, such as the Datagram Congestion Control Protocol (DCCP) [[RFC4340](#)].

<foundation>: 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, and is composed of one to thirty two <ice-char>. The foundation is used to optimize ICE performance in the Frozen algorithm.

<component-id>: identifies the specific component of the media stream for which this is a candidate and as a positive integer between 1 and 256. 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. Other types of media streams which require multiple components MUST develop specifications which define the mapping of components to component IDs. See [Section 14](#) 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. The ICE 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> is 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 [Appendix B.3](#) of ICE [[RFC5245](#)] for a discussion of its purpose). If the candidate is a host candidate <rel-addr> and <rel-port> MUST be omitted.

<tcp-type-ext>: conveys the candidates connection type (active, passive, or S-O) for TCP based candidates. This MUST be included for candidates that has <transport> set to TCP and MUST NOT be included for other transport types, including UDP, unless explicitly specified for that transport protocol.

4.3. ICE Password and Username Transport Header Parameters

The ICE password and username for each agent needs to be transported using RTSP. For that purpose new transport header parameters are defined.

There MUST be an "ICE-Password" and "ICE-ufrag" parameter for each media stream. If two SETUP requests in the same RTSP session have identical ICE-ufrag's, they MUST have identical ICE-Password's. The ICE-ufrag and ICE-Password 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-Password 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-Password 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.

The ABNF [[RFC5234](#)] for these parameters are:

```
trns-parameter    =/ SEMI ice-password-par
trns-parameter    =/ SEMI ice-ufrag-par
ice-password-par  = "ICE-Password" EQUAL DQ password DQ
ice-ufrag-par     = "ICE-ufrag" EQUAL DQ ufrag DQ
password          = <Defined in [RFC5245]>
ufrag             = <Defined in [RFC5245]>
EQUAL            = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
SEMI             = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
DQ               = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
```

4.4. ICE Feature Tag

A feature tag is defined for use in the RTSP capabilities mechanism for ICE support of media transport using datagrams: "setup.ice-d-m". This feature tag indicates that one supports all the mandatory functions of this specification. It is applicable to all types of

RTSP agents; clients, servers and proxies.

The RTSP client SHOULD send the feature tag "setup.ice-d-m" in the "Supported" header in all SETUP requests that contain the "D-ICE" lower layer transport.

4.5. Status Codes

ICE needs two new RTSP response codes to indicate correctly progress and errors.

Code	Reason	Method
150	Server still working on ICE connectivity checks	PLAY
480	ICE Connectivity check failure	PLAY, SETUP

Table 1: New Status codes and their usage with RTSP methods

4.5.1. 150 ICE connectivity checks in progress

The 150 response code indicates that ICE connectivity checks are still in progress and haven't concluded. This response SHALL be sent within 200 milliseconds of receiving a PLAY request that currently can't be fulfilled because ICE connectivity checks are still running. Subsequently, every 3 seconds after the previous sent one, a 150 reply shall be sent until the ICE connectivity checks conclude either successfully or in failure, and a final response for the request can be provided.

4.5.2. 480 ICE Processing Failed

The 480 client error response code is used in cases when the request can't be fulfilled due to a failure in the ICE processing, such as that all the connectivity checks have timed out. This error message can appear either in response to a SETUP request to indicate that no candidate pair can be constructed or to a PLAY request that the server's connectivity checks resulted in failure.

4.6. New Reason for PLAY_NOTIFY

A new value used in the PLAY_NOTIFY methods Notify-Reason header is defined: "ice-restart". This reason indicates that a ICE restart needs to happen on the identified resource and session.

Notify-Reas-val =/ "ice-restart"

4.7. Server Side SDP Attribute for ICE Support

If the server supports the media NAT traversal for RTSP controlled sessions, as described in this RFC, then the Server SHOULD include the "a=rtsp-ice-d-m" SDP attribute in any SDP (if used) describing content served by the server. This is an session level attribute.

```
rtsp-ice-d-m-attr = "a=" "rtsp-ice-d-m"
```

4.8. ICE Features Not Required in RTSP

A number of ICE signalling features are not needed with RTSP and are discussed below.

4.8.1. ICE-Lite

The ICE-Lite attribute shall not be used in the context of RTSP. The ICE specification describes two implementations of ICE: Full and Lite, where hosts that are not behind a NAT are allowed to implement only Lite. For RTSP, the Lite implementation is insufficient because it does not cause the media server to send a connectivity check, which are used to protect against making the RTSP server a denial of service tool. This document defines another variation implementation of ICE, called ICE-RTSP. It has its own set of simplifications suitable to RTSP. Conceptually, this implementation of ICE-RTSP is between ICE-FULL and ICE-LITE for a server and simpler than ICE-FULL for clients.

4.8.2. ICE-Mismatch

The ice-mismatch parameter indicates that the offer arrived with a default destination for a media component that didn't have a corresponding candidate attribute. This is not needed for RTSP as the ICE based lower layer transport specification either is supported or another alternative transport is used. This is always explicitly indicated in the SETUP request and response.

4.8.3. ICE Remote Candidate Transport Header Parameter

The Remote candidate attribute is not needed for RTSP for the following reasons. Each SETUP results in a independent ICE processing chain which either fails or results in promoting a single candidate pair to usage. If a new SETUP request for the same media is sent this needs to use a new userfragment and password to avoid any race conditions or uncertainty for which processing round the STUN requests relate to.

5. Detailed Solution

This section describes in detail how the interaction and flow of ICE works with RTSP messages.

5.1. Session description and RTSP DESCRIBE (optional)

The RTSP server should indicate it has support for ICE by sending the "a=rtsp-ice-d-m" SDP attribute in the response to the RTSP DESCRIBE message if SDP is used. This allows RTSP clients to only send the new ICE interchanges with servers that support ICE so limiting the overhead on current non-ICE supporting RTSP servers. When not using RTSP DESCRIBE it is still recommended to use the SDP attribute for session description.

A Client can also use the DESCRIBE request to determine explicitly if both server and any proxies support ICE. The client includes the "Supported" header with its supported feature tags, including "setup.ice-d-m". Any proxy upon seeing the "Supported" header will include the "Proxy-Supported" header with the feature tags it supports. The server will echo back the "Proxy-Supported" header and its own version of the Supported header so enabling a client to determine if all involved parties support ICE or not. Note that even if a proxy is present in the chain that doesn't indicate support for ICE, it may still work.

For example:

```
C->S: DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/2.0
      CSeq: 312
      User-Agent: PhonyClient 1.2
      Accept: application/sdp, application/example
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

```
S->C: RTSP/2.0 200 OK
      CSeq: 312
      Date: 23 Jan 1997 15:35:06 GMT
      Server: PhonyServer 1.1
      Content-Type: application/sdp
      Content-Length: 367
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 192.0.2.46
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/lectures/sdp.ps
e=seminar@example.com (Seminar Management)
t=2873397496 2873404696
a=recvonly
a=rtsp-ice-d-m
a=control: *
m=audio 3456 RTP/AVP 0
a=control: /audio
m=video 2232 RTP/AVP 31
a=control: /video
```

5.2. Setting up the Media Streams

The RTSP client reviews the session description returned, for example by an RTSP DESCRIBE message, to determine what media resources that need to be setup. For each of these media streams where the transport protocol supports ICE connectivity checks, the client SHALL gather candidate addresses for UDP transport as described in [section 4.1.1](#) in ICE [[RFC5245](#)] according to standard ICE rather than the ICE-Lite implementation and according to [section 5](#) of ICE TCP [[RFC6544](#)] for TCP based candidates.

5.3. RTSP SETUP Request

The RTSP client will then send at least one SETUP request per media stream to establish the media streams required for the desired session. For each media stream where it desires to use ICE it will include a transport specification with "D-ICE" as the lower layer, and each media stream SHALL have its own unique combination of ICE

candidates and ICE-ufrag. This transport specification SHOULD be placed first in the list to give it highest priority. It is RECOMMENDED that additional transport specifications are provided as a fallback in case of non ICE supporting proxies. The RTSP client will be initiating and thus the controlling party in the ICE processing. For example (Note that some lines are broken in contradiction with the defined syntax due to space restrictions in the documenting format:

```
C->S: SETUP rtsp://server.example.com/fizzle/foo/audio RTSP/2.0
      CSeq: 313
      Transport: RTP/AVP/D-ICE; unicast; ICE-ufrag=8hhY;
                 ICE-Password=asd88fgpdd777uzjYhagZg; candidates="
                 1 1 UDP 2130706431 10.0.1.17 8998 typ host;
                 2 1 UDP 1694498815 192.0.2.3 45664 typ srflx
                   raddr 10.0.1.17 rport 8998"; RTCP-mux,
                 RTP/AVP/UDP; unicast; dest_addr=":6970"/":6971",
                 RTP/AVP/TCP;unicast;interleaved=0-1
      Accept-Ranges: NPT, UTC
      User-Agent: PhonyClient/1.2
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

5.4. Gathering Candidates

Upon receiving a SETUP request the server can determine what media resource should be delivered and which transport alternatives that the client supports. If one based on D-ICE is on the list of supported transports and preferred among the supported, the below applies.

The transport specification will provide which media protocol is to be used and based on this and the clients candidates, the server determines the protocol and if it supports ICE with that protocol. The server shall then gather its UDP candidates according to [section 4.1.1](#) in ICE [[RFC5245](#)] and any TCP based ones according to [section 5](#) of ICE TCP [[RFC6544](#)].

Servers that have an address that is generally reachable by any clients within the address scope the server intends to serve MAY be specially configured (high-reachability configuration). This special configuration has the goal of reducing the server side candidate to preferably a single one per (address family, media stream, media component) tuple. Instead of gathering all possible addresses including relayed and server reflexive addresses, the server uses a single address per address family that it knows it should be reachable by a client behind one or more NATs. The reason for this special configuration is two fold: Firstly it reduces the load on the server in address gathering and in ICE processing during the

connectivity checks. Secondly it will reduce the number of permutations for candidate pairs significantly thus potentially speeding up the conclusion of the ICE processing. Note however that using this option on a server that doesn't fulfill the requirement of being reachable is counter-productive and it is important that this is correctly configured.

The above general consideration for servers applies also for TCP based candidates. A general implementation should support several candidate collection techniques and connection types. For TCP based candidates a high-reachability configured server is recommended to only offer Host candidates. In addition to passive connection types the server can select to provide active or SO conenction types to match the client's candidates.

5.5. RTSP Server Response

The server determines if the SETUP request is successful from the other perspectives and will return a 200 OK response, otherwise returning an error code. At that point the server, having selected a transport specification using the "D-ICE" lower layer, will need to include that transport specification in the response message. The transport specification shall include the candidates gathered in [Section 5.4](#) in the "candidates" transport header parameter as well as the server's username and password fragments. In the case that there are no valid candidate pairs with the combination of the client and servers candidates, a 480 (ICE Processing Failed) error response SHALL be returned which MUST include the servers' candidates. The return of a 480 error allows both the server and client to release its candidates.

Example of a succesful response to the request in [Section 5.3](#).

```
S->C: RTSP/2.0 200 OK
      CSeq: 313
      Session: 12345678
      Transport: RTP/AVP/D-ICE; unicast; RTCP-mux; ICE-ufrag=MkQ3;
                 ICE-Password=pos12Dgp9FcAjpq82ppaF; candidates="
                 1 1 UDP 2130706431 192.0.2.56 50234 typ host"
      Accept-Ranges: NPT
      Date: 23 Jan 1997 15:35:06 GMT
      Server: PhonyServer 1.1
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```


5.6. Server to Client ICE Connectivity Checks

The server shall start the connectivity checks following the procedures described in [Section 5.7](#) and 5.8 of ICE [[RFC5245](#)] unless it is configured to use the high-reachability option. If it is then it MAY suppress its own checks until the servers checks are triggered by the client's connectivity checks.

Please note that [section 5.8](#) does specify that the initiation of the checks are paced and new ones are only started every T_a milliseconds. The motivation for this is documented in [Appendix B.1](#) of ICE [[RFC5245](#)] as for SIP/SDP all media streams within an offer/answer dialog are running using the same queue. To ensure the same behavior with RTSP, the server SHALL use a single pacer queue for all media streams within each RTSP session.

The values for the pacing of STUN and TURN transactions T_a and T_0 can be configured but have the same minimum values defined in the ICE specification.

When a connectivity check from the client reaches the server it will result in a triggered check from the server as specified in [section 7.2.1.4](#) of ICE [[RFC5245](#)]. This is why servers with a high reachability address can wait until this triggered check to send out any checks for itself so saving resources and mitigating the DDoS potential.

5.7. Client to Server ICE Connectivity Check

The client receives the SETUP response and learns the candidate address to use for the connectivity checks. The client shall initiate its connectivity check, following the procedures in [Section 6](#) of ICE [[RFC5245](#)]. The STUN transaction pacer SHALL be used across all media streams part of the same RTSP session.

Aggressive nomination SHALL be used with RTSP. This doesn't have the negative impact that it has in offer/answer as media playing only starts after issuing a PLAY request.

5.8. Client Connectivity Checks Complete

When the client has concluded all of its connectivity checks and has nominated its desired candidate for a particular media stream, it MAY issue a PLAY request for that stream. Note, that due to the aggressive nomination, there is a risk that any outstanding check may nominate another pair than what was already nominated. If the client has locally determined that its checks have failed it may try providing an extended set of candidates and update the server

candidate list by issuing a new SETUP request for the media stream.

If the client concluded its connectivity checks successfully and therefore sent a PLAY request but the server cannot conclude successfully, the server will respond with a 480 (ICE Processing Failed). Upon receiving the 480 (ICE Processing Failed) response, the client may send a new SETUP request assuming it has any new information that can be included in the candidate list. If the server is still performing the checks when receiving the PLAY request it will respond with a 150 (ICE connectivity checks in progress) response to indicate this.

5.9. Server Connectivity Checks Complete

When the RTSP server receives a PLAY request, it checks to see that the connectivity checks have concluded successfully and only then will it play the stream. If the PLAY request is for a particular media stream, the server only needs to check that the connectivity checks for that stream completely successfully. If the server has not concluded its connectivity checks the server indicates that by sending the 150 (ICE connectivity checks in progress) ([Section 4.5.1](#)). If there is a problem with the checks then the server sends to the client a 480 response to indicate a failure of the checks. If the checks are successful then the server sends a 200 OK response and starts delivering media.

5.10. Releasing Candidates

Both server and client MAY release its non nominated candidates as soon as a 200 PLAY response has been issued/received and no outstanding connectivity checks exist.

5.11. Steady State

The client and server SHALL use STUN to send keep-alive for the nominated candidate pair(s) following the rules of [Section 10](#) of ICE [[RFC5245](#)]. This is important as normally RTSP play mode sessions only contain traffic from the server to the client so the bindings in the NAT need to be refreshed by the client to server traffic provided by the STUN keep-alive.

5.12. Re-SETUP

The server SHALL support SETUP requests in PLAYING state, as long as the SETUP changes only the ICE parameters, which are: ICE-Password, ICE-ufrag and the content of ICE candidates.

If the client decides to change any parameters related to the media

stream setup it will send a new SETUP request. In this new SETUP request the client MAY include a new different username and password to use in the ICE processing. New username and passwords SHALL cause the ICE processing to start from the beginning again, i.e. an ICE restart. The client SHALL in case of ICE restart gather candidates and include the candidates in the transport specification for D-ICE.

If the RTSP session is in playing state at the time of sending the SETUP request requiring ICE restart, then the ICE connectivity checks SHALL use Regular nomination. Any ongoing media delivery continues on the previously nominated candidate pairs until the new pairs have been nominated for the individual candidate. Once the nomination of the new candidate pair has completed, all unused candidates may be released.

5.13. Server Side Changes After Steady State

A Server may require an ICE restart because of server side load balancing or a failure resulting in an IP address and a port number change. It shall use the PLAY_NOTIFY method to inform the client ([Section 13.5 \[I-D.ietf-mmusic-rfc2326bis\]](#)) with a new Notify-Reason header: ice-restart. The server will identify if the change is for a single media or for the complete session by including the corresponding URI in the PLAY_NOTIFY request.

Upon receiving and responding to this PLAY_NOTIFY with ice-restart reason the client SHALL gather new ICE candidates, send SETUP requests for each media stream part of the session. The server provides its candidates in the SETUP response the same way as for the first time ICE processing. Both server and client shall provide new ICE usernames and passwords. The client MAY issue the SETUP request while the session is in PLAYING state.

If the RTSP session is in PLAYING state when the client issues the SETUP request, the client SHALL use regular nomination. If not the client will use the same procedures as for when first creating the session.

Note that keepalives on the previous set of candidate pairs should continue until all new candidate pairs have been nominated. After having nominated a new set of candidate pairs, the client may continue to receive media for some additional time. Even if the server stops delivering media over that candidate pair at the time of nomination, media may arrive for up to one maximum segment lifetime as defined in TCP (2 minutes). Unfortunately, if the RTSP server is divided into a separate controller and media streamer, a failure may result in continued media delivery for a longer time than the maximum segment lifetime, thus source filtering is recommended.

For example:

```
S->C: PLAY_NOTIFY rtsp://example.com/fizzle/foo RTSP/2.0
      CSeq: 854
      Notify-Reason: ice-restart
      Session: uZ3ci0K+Ld
      Server: PhonyServer 1.1

C->S: RTSP/2.0 200 OK
      CSeq: 854
      User-Agent: PhonyClient/1.2

C->S: SETUP rtsp://server.example.com/fizzle/foo/audio RTSP/2.0
      CSeq: 314
      Session: uZ3ci0K+Ld
      Transport: RTP/AVP/D-ICE; unicast; ICE-ufrag=Kl1C;
                  ICE-Password=H4sICGjBsEcCA3Rlc3RzLX; candidates="
                  1 1 UDP 2130706431 10.0.1.17 8998 typ host;
                  2 1 UDP 1694498815 192.0.2.3 51456 typ srflx
                      raddr 10.0.1.17 rport 9002"; RTCP-mux,
                  RTP/AVP/UDP; unicast; dest_addr=":6970"/":6971",
                  RTP/AVP/TCP;unicast;interleaved=0-1
      Accept-Ranges: NPT, UTC
      User-Agent: PhonyClient/1.2

C->S: SETUP rtsp://server.example.com/fizzle/foo/video RTSP/2.0
      CSeq: 315
      Session: uZ3ci0K+Ld
      Transport: RTP/AVP/D-ICE; unicast; ICE-ufrag=hZv9;
                  ICE-Password=JAhA9myMHETTFNCrPtg+kJ; candidates="
                  1 1 UDP 2130706431 10.0.1.17 9000 typ host;
                  2 1 UDP 1694498815 192.0.2.3 51576 typ srflx
                      raddr 10.0.1.17 rport 9000"; RTCP-mux,
                  RTP/AVP/UDP; unicast; dest_addr=":6972"/":6973",
                  RTP/AVP/TCP;unicast;interleaved=0-1
      Accept-Ranges: NPT, UTC
      User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
      CSeq: 314
      Session: uZ3ci0K+Ld
      Transport: RTP/AVP/D-ICE; unicast; RTCP-mux; ICE-ufrag=CbDm;
                  ICE-Password=OfdXHws9XX0eBr6j2zz9Ak; candidates="
                  1 1 UDP 2130706431 192.0.2.56 50234 typ host"
      Accept-Ranges: NPT
      Date: 11 March 2011 13:17:46 GMT
      Server: PhonyServer 1.1
```



```
S->C: RTSP/2.0 200 OK
      CSeq: 315
      Session: uZ3ci0K+Ld
      Transport: RTP/AVP/D-ICE; unicast; RTCP-mux; ICE-ufrag=jigs;
                ICE-Password=Dgx6fPj2lsa2WI8b7oJ7+s; candidates="
                1 1 UDP 2130706431 192.0.2.56 47233 typ host"
      Accept-Ranges: NPT
      Date: 11 March 2011 13:17:47 GMT
      Server: PhonyServer 1.1
```

6. ICE and Proxies

RTSP allows for proxies which can be of two fundamental types depending if they relay and potentially cache the media or not. Their differing impact on the RTSP NAT traversal solution, including backwards compatibility, is explained below.

6.1. Media Handling Proxies

An RTSP proxy that relays or caches the media stream for a particular media session can be considered to split the media transport into two parts: A media transport between the server and the proxy according to the proxies need, and delivery from the proxy to the client. This split means that the NAT traversal solution will need to be run on each individual media leg according to need.

It is RECOMMENDED that any media handling proxy support the media NAT traversal defined within this specification. This is for two reasons: Firstly to enable clients to perform NAT traversal for the media between the proxy and itself and secondly to allow the proxy to be topology independent so able to support performing NAT traversal for non-NAT traversal capable clients present in the same address domain.

For a proxy to support the media NAT traversal defined in this specification a proxy will need to implement the solution fully and be ready as both a controlling and a controlled ICE peer. The proxy also SHALL include the "setup.ice-d-m" feature tag in any applicable capability negotiation headers, such as "Proxy-Supported."

6.2. Signalling Only Proxies

A signalling only proxy handles only the RTSP signalling and does not have the media relayed through proxy functions. This type of proxy is not likely to work unless the media NAT traversal solution is in place between the client and the server, because the DoS protection measures usually prevent media delivery to other addresses other than

from where the RTSP signalling arrives at the server.

The solution for the Signalling Only proxy is that it must forward the RTSP SETUP requests including any transport specification with the "D-ICE" lower layer and the related transport parameters. A proxy supporting this functionality SHOULD indicate its capability by always including the "setup.ice-d-m" feature tag in the "Proxy-Supported" header.

6.3. Non-supporting Proxies

A media handling proxy that doesn't support the ICE media NAT traversal specified here is assumed to remove the transport specification and use any of the lower prioritized transport specifications if provided by the requester. The specification of such a non ICE transport enables the negotiation to complete, although with a less preferred method as a NAT between the proxy and the client will likely result in failure of the media path.

A non-media handling proxy is expected to ignore and simply forward all unknown transport specifications, however, this can only be guaranteed for proxies following the published RTSP 2.0 specification.

Unfortunately the usage of the "setup.ice-d-m" feature tag in the Proxy-Require will have contradicting results. For a non ICE supporting media handling proxy, the inclusion of the feature tag will result in aborting the setup and indicating that it isn't supported, which is desirable if you want to provide other fallbacks or other transport configurations to handle the situation. For non-supporting non-media handling proxies the result will also result in aborting the setup, however, setup might have worked if the proxy-require tag wasn't present. This variance in results is the reason we don't recommend the usage of the Proxy-Require header. Instead we recommend the usage of the Supported header to force proxies to include the feature tags they support in the proxy-supported which will provide a positive indication when all proxies in the chain between the client and server support the functionality. Even if not explicitly indicating support, any SETUP response including a transport specification with "D-ICE" will be implicit indication that the proxy chain supports at least passthrough of this media.

7. RTP and RTCP Multiplexing

"Multiplexing RTP Data and Control Packets on a Single Port" [[RFC5761](#)] specifies how and when RTP and RTCP can be multiplexed on the same port. This multiplexing SHALL be combined with ICE as it

makes RTP and RTCP need only a single component per media stream instead of two, so reducing the load on the connectivity checks. For details on how one negotiate RTP and RTCP multiplexing, see [Appendix B](#) of RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)].

Multiplexing RTP and RTCP has the benefit that it avoids the need for handling two components per media stream when RTP is used as the media transport protocol. This eliminates at least one STUN check per media stream and will also reduce the time needed to complete the ICE processing by at least the time it takes to pace out the additional STUN checks of up to one complete round trip time for a single media stream. In addition to the protocol performance improvements, the server and client side complexities are reduced as multiplexing halves the total number of STUN instances and holding the associate state. Multiplexing will also reduce the combinations and length of the list of possible candidates.

The implementation of RTP and RTCP multiplexing is additional work required for this solution. However, when implementing the ICE solution a server or client will need to implement a de-multiplexer between the STUN, and RTP or RTCP packets below the RTP/RTCP implementation anyway, so the additional work of one new demultiplexing point directly connected to the STUN and RTP/RTCP seems small relative to the benefits provided.

Due to the above mentioned benefits, RTSP servers and clients that supports "D-ICE" lower layer transport in combination with RTP SHALL also implement RTP and RTCP multiplexing as specified in this section and [[RFC5761](#)].

8. Fallback

The need for fallback from ICE in RTSP should be less than for SIP using ICE in SDP offer/answer where a default destination candidate is very important. This as capability determination for ICE can happen prior to the RTSP SETUP request. Thus a client should normally not needed to include fallback alternatives when offering ICE, as the capability for ICE will already be determined. Thus this section likely applies more to the cases where the server is not ICE capable and the client wishes to use part of the ICE functionality to improve NAT/Firewall traversal.

[Section 4.1.4](#) of the ICE [[RFC5245](#)] specification does recommend that the default destination, i.e. what is used in fallback if the peer isn't ICE capable is a candidate of relayed type to maximize the likelihood of succesfull transport of media. This is based on that the peer in SIP SDP offer/answer is almost as likely as the RTSP

client to be behind a NAT. For RTSP the deployment of servers are much more heavily weighted towards deployment with public reachability. In fact due to that servers behind NAT either needs to support ICE or have static configurations that allow traversal one can assume that the server will have a public address or support ICE. Thus, the selection of the default destination address for RTSP can be differently prioritized.

As an ICE enabled client needs to be configured with a STUN server address to be able to gather candidates successfully, this can be utilized to derive a server reflexive candidate for the clients port. How useful this is for an RTSP client as default candidate depends on the properties of the NAT. As long as the NAT use an address independent mapping, then using a STUN derived reflexive candidate is likely to be successfully. This is however brittle in several ways. First, the NATs behavior can be determined using STUN as described in [\[RFC3489\]](#), however this might not be representative of the behavior encountered in another mapping. Secondly, filter state towards the ports used by the server needs to be established. This requires that the server actually include both address and ports in its response to the SETUP request. Thirdly messages needs to be sent to these ports for keep-alive at a regular interval. How a server reacts to such unsolicited traffic is unknown. This brittleness may be accepted in fallback due to lack of support on the server side.

Fallback addresses needs to be provided in their own transport specification using a specifier that do not include the "D-ICE" lower layer transport. Instead the selected protocol, e.g. UDP needs to be explicitly or implicitly indicated. Secondly the selected default candidate needs to be included in the SETUP request. If this candidate is server reflexive or relayed the aspect of keep-alive needs to be ensured.

9. IANA Considerations

This document request registration in a number of registries, both for RTSP and SDP.

9.1. RTSP Feature Tags

This document request that one RTSP 2.0 feature tags are registered in the "RTSP 2.0 Feature-tags" registry:

setup.ice-d-m See [Section 4.4](#). This feature tag applies to clients, servers and proxies.

9.2. Transport Protocol Specifications

This document needs to register a number of transport protocol combinations are registered in RTSP 2.0's "Transport Protocol Specifications" registry.

"RTP/AVP/D-ICE" RTP using the AVP profile over an ICE established datagram flow.

"RTP/AVPF/D-ICE" RTP using the AVPF profile over an ICE established datagram flow.

"RTP/SAVP/D-ICE" RTP using the SAVP profile over an ICE established datagram flow.

"RTP/SAVPF/D-ICE" RTP using the SAVPF profile over an ICE established datagram flow.

9.3. RTSP Transport Parameters

This document requests that 3 transport parameters are registered in RTSP 2.0's "Transport Parameters":

"candidates": Listing the properties of one or more ICE candidate. See Section [Section 4.2](#).

"ICE-Password": The ICE password used to authenticate the STUN binding request in the ICE connectivity checks. See Section [Section 4.3](#).

"ICE-ufrag": The ICE username fragment used to authenticate the STUN binding requests in the ICE connectivity checks. See Section [Section 4.3](#).

9.4. RTSP Status Codes

This document requests that 2 assignments are done in the "RTSP 2.0 Status Codes" registry. The suggested values are:

150: See Section [Section 4.5.1](#).

480: See Section [Section 4.5.2](#).

9.5. Notify-Reason value

This document requests that one assignment is done in the RTSP 2.0 Notify-Reason header value registry. The defined value is:

ice-restart: Server notifying the client about the need for an ICE restart. See section [Section 4.6](#).

9.6. SDP Attribute

The registration of one SDP attribute is requested:
SDP Attribute ("att-field"):

Attribute name:	rtsp-ice-d-m
Long form:	ICE for RTSP datagram media NAT traversal
Type of name:	att-field
Type of attribute:	Session level only
Subject to charset:	No
Purpose:	RFC XXXX
Reference:	RFC XXXX
Values:	No values defined.
Contact:	Magnus Westerlund
	E-mail: magnus.westerlund@ericsson.com
	phone: +46 10 714 82 87

10. Security Considerations

ICE [[RFC5245](#)] and ICE TCP [[RFC6544](#)] provides an extensive discussion on security considerations which applies here as well.

10.1. ICE and RTSP

A long-standing risk with transmitting a packet stream over UDP is that the host may not be interested in receiving the stream. On today's Internet many hosts are behind NATs or operate host firewalls which do not respond to unsolicited packets with an ICMP port unreachable error. Thus, an attacker can construct RTSP SETUP requests with a victim's IP address and cause a flood of media packets to be sent to a victim. The addition of ICE, as described in this document, provides protection from the attack described above. By performing the ICE connectivity check, the media server receives confirmation that the RTSP client wants the media. While this protection could also be implemented by requiring the IP addresses in the SDP match the IP address of the RTSP signaling packet, such a mechanism does not protect other hosts with the same IP address (such as behind the same NAT), and such a mechanism would prohibit separating the RTSP controller from the media playout device (e.g.,

an IP-enabled remote control and an IP-enabled television), it also forces RTSP proxies to relay the media streams through them, even if they only are signalling proxies.

To protect against the attacks in ICE based on signalling information RTSP signalling should be protected using TLS to prevent eavesdropping and modification of information.

The STUN amplification attack described in [Section 18.5.2](#) in ICE needs consideration. For servers that are able to run according to the high-reachability option have good mitigation against this attack as it only sends its connectivity checks towards an address and port pair it has received an incoming connectivity check from. Thus requiring the capability to spoof source addresses in addition to signal the RTSP server a set of ICE candidates. Independently an ICE agent needs to implement the mitigation to reduce the volume of the amplification attack as described in the ICE specification.

[11.](#) Acknowledgements

The authors would like to thank Remi Denis-Courmont for suggesting the method of integrating ICE in RTSP signalling, Dan Wing for help with the security section and numerous other issues.

[12.](#) References

[12.1.](#) Normative References

- [I-D.ietf-mmusic-rfc2326bis]
Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M.,
and M. Stiemerling, "Real Time Streaming Protocol 2.0
(RTSP)", [draft-ietf-mmusic-rfc2326bis-29](#) (work in
progress), March 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
Description Protocol", [RFC 4566](#), July 2006.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
Specifications: ABNF", STD 68, [RFC 5234](#), January 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment
(ICE): A Protocol for Network Address Translator (NAT)
Traversal for Offer/Answer Protocols", [RFC 5245](#),

April 2010.

- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", [RFC 5389](#), October 2008.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", [RFC 5761](#), April 2010.
- [RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", [RFC 6544](#), March 2012.

12.2. Informative References

- [I-D.ietf-mmusic-rtsp-nat-evaluation]
Westerlund, M. and T. Zeng, "The Evaluation of Different Network Address Translator (NAT) Traversal Techniques for Media Controlled by Real-time Streaming Protocol (RTSP)", [draft-ietf-mmusic-rtsp-nat-evaluation-04](#) (work in progress), October 2011.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", [RFC 2326](#), April 1998.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", [RFC 3022](#), January 2001.
- [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.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", [RFC 4340](#), March 2006.

Authors' Addresses

Jeff Goldberg
Cisco
11 New Square, Bedfont Lakes
Feltham,, Middx TW14 8HA
United Kingdom

Phone: +44 20 8824 1000
Fax:
Email: jgoldber@cisco.com
URI:

Magnus Westerlund
Ericsson
Torshamsgatan 23
Stockholm, SE-164 80
Sweden

Phone: +46 8 719 0000
Fax:
Email: magnus.westerlund@ericsson.com
URI:

Thomas Zeng
Nextwave Wireless, Inc.
12670 High Bluff Drive
San Diego, CA 92130
USA

Phone: +1 858 480 3100
Fax:
Email: thomas.zeng@gmail.com
URI:

