Internet Engineering Task Force (IETF)
Request for Comments: 7825
Category: Standards Track
ISSN: 2070-1721

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

December 2016

A Network Address Translator (NAT) Traversal Mechanism for Media Controlled by the Real-Time Streaming Protocol (RTSP)

Abstract

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

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7825.

Table of Contents

1. Introduction

"Real Time Streaming Protocol (RTSP)" [RFC2326] and RTSP 2.0 [RFC7826] are protocols used to set up and control one or more media streams delivering media to receivers. It is RTSP's functionality of setting up media streams that causes serious issues with Network Address Translators (NATs) [RFC3022] unless extra provisions are made by the protocol. Thus, there is 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 [RFC7826]. RTSP 2.0 is similar to RTSP 1.0 in many respects, but, significantly for this work, it contains a well-defined extension mechanism that allows 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 and cause compatibility issues.

There have been a number of suggested ways of resolving the NAT traversal of media for RTSP, most of which are already used in implementations. The evaluation of these NAT-traversal solutions in [RFC7604] has shown that there are many issues to consider. After extensive evaluation, a mechanism based on Interactive Connectivity Establishment (ICE) [RFC5245] was selected. There were mainly two reasons: the mechanism supports RTSP servers behind NATs and 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 from server to client. If future extensions are specified for other delivery modes than "PLAY", then the optimizations in regard to when PLAY requests are sent needs to be reconsidered.

The NAT problem for RTSP signaling traffic is a less prevalent problem than the NAT problem for RTSP media streams. Consequently, the former is left for future study.

The ICE usage defined in this specification is called "ICE-RTSP" and does not match the full ICE for SIP/SDP (Session Description Protocol) or ICE-Lite as defined in the ICE specification [RFC5245]. ICE-RTSP is tailored to the needs of RTSP and is slightly simpler than ICE-Full for both clients and servers.

2. Key Words

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].

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. The RTSP server should indicate it has support for ICE via a new SDP [RFC4566] attribute ("a=rtsp-ice-d-m") in, for example, the SDP returned in the RTSP DESCRIBE message. This allows RTSP clients to only perform the new ICE exchanges with servers that support ICE. If RTSP DESCRIBE is used, the normal capability determination mechanism should also be used, i.e., Supported

header with a new ICE feature tag. Note: both mechanisms should be supported, as there are various use cases where only one of them is 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 set up. For each of these media streams where the transport protocol supports connectivity checks based on Session Traversal Utilities for (NAT) (STUN) [RFC5389], the client gathers candidate addresses. See Section 4.1.1 in ICE [RFC5245]. The client then runs a STUN server on each of the local candidate's transport addresses it has gathered.

3. The RTSP client sends SETUP requests containing 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 is not behind a NAT, 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 Sections 5.7 and 5.8 of ICE [RFC5245]. If the server is not behind a NAT and uses a public IP address with a single candidate per (media stream, component, address family) tuple, then the server may be configured to not initiate connectivity checks.

7. The client receives the SETUP response and learns the candidate addresses 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 not behind a NAT can wait until this triggered check to send out any checks for itself, so saving resources and mitigating the DDoS potential from server-initiated connectivity checks.

9. When the client has concluded its connectivity checks, including nominating candidates, and has correspondingly received the server connectivity checks on the nominated 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 if 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 server receives a PLAY request, it checks to see that the connectivity checks have concluded successfully, and only then can it play the stream. If there is a problem with the checks, then the server sends either a 150 (Server still working on ICE connectivity checks) response to show that it is still working on the connectivity checks, or a 480 (ICE Connectivity check failure) 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, a single candidate per component has been nominated, and a PLAY response has been received (client) or sent (server).

The client needs to continue to use STUN as a keep-alive mechanism for the used candidate pairs to keep their NAT bindings current. RTSP servers behind NATs will also need to send keep-alive messages when not sending media. This is important since RTSP media sessions often contain only media traffic from the server to the client so the bindings in the NAT need to be refreshed by 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 otherwise.

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 possible to demultiplex from STUN messages (see STUN [RFC5389]). By "datagram clean" we mean that it has to 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

that 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 that use datagrams. This specification defines the following combinations:

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

This list can be extended with more transport specifications after having performed the evaluation that they are compatible with D-ICE as lower layer. The registration is required to follow the registry rules for the Transport Protocol Identifier (see Section 22.13.1 of [RFC7826]).

The lower-layer "D-ICE" has the following rules for the inclusion of the RTSP Transport header (Section 18.54 of RTSP 2.0 [RFC7826]) parameters:

unicast: ICE only supports unicast operations; thus, it is REQUIRED that one include the unicast indicator parameter (see Section 18.54 in RTSP 2.0 [RFC7826]).

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

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

ICE-Password: This parameter SHALL be included (see Section 4.2).

ICE-ufrag: This parameter SHALL be included (see Section 4.2).

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 specifications using "D-ICE" as lower layer, this parameter MUST be present. The parameter can contain one or more ICE candidates. In the SETUP response, there is only a single transport specification; if that uses the "D-ICE" lower layer, this parameter MUST be present and include the server-side candidates.

The ABNF [RFC5234] for these transport header parameters are:

```
trns-parameter = <Defined in Section 20.2.3 of [RFC7826]>
trns-parameter =/ SEMI ice-trn-par
ice-trn-par    = "candidates" EQUAL DQUOTE SWS ice-candidate
                                    *(SEMI ice-candidate) SWS DQUOTE
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]>
priority              = <See Section 15.1 of [RFC5245]>
cand-type             = <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]>
connection-address    = <See [RFC4566]>
port                  = <See [RFC4566]>
EQUAL                 = <Defined in [RFC7826]>
```

```
DQUOTE                = <Defined in [RFC7826]>
SWS                   = <Defined in [RFC7826]>
SEMI                  = <Defined in [RFC7826]>
SP                    = <Defined in [RFC7826]>
```

<connection-address>: is the unicast IP address of the candidate, allowing for IPv4 addresses, IPv6 addresses, and Fully Qualified Domain Names (FQDNs), taken from SDP [RFC4566]. Note, this context MUST have a unicast address for this parameter, even though a multicast address would be syntactically valid. The connection address SHOULD use the same format (explicit IP or FQDN) as in the dest_addr parameter used in the transport specification that express any fallback. An IP address is preferred for simplicity, but both an IP Address and FQDN can be used. In the FQDN case, when receiving a SETUP request or response containing an FQDN in an ice-candidate parameter, the FQDN is looked up in the DNS first using a 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 is 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 IP address, and come from the same STUN server. It is composed of one to thirty two <ice-char>. The foundation is used to optimize ICE performance in the Frozen algorithm (as described in [RFC5245]).

<component-id>: identifies the specific component of the media stream for which this is a candidate and is a positive integer belonging to the range 1-256. It MUST start at 1 and MUST increment by 1 for each component of a particular media stream. 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 unless RTP and RTCP Multiplexing

(Section 8) is used, in which case the second component is omitted and RTP and RTCP are both transported over the first component. Other types of media streams that require multiple components MUST develop specifications that define the mapping of components to component IDs. See Section 14 in [RFC5245] for additional discussion on extending ICE to new media streams.

<priority>: is a positive integer in the range 1 to (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> are equal to the base for that server- or peer-reflexive candidate. If the candidate is relayed, <rel-addr> and <rel-port> are equal to the mapped address in the TURN 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 candidate's connection type (active, passive, or simultaneous-open (S-O)) for TCP-based candidates. This MUST be included for candidates that have <transport> set to TCP and MUST NOT be included for other transport types, including UDP.

<extension-att-name> and <extension-att-value>: These are prototypes for future extensions of the candidate line. The ABNF for these allows any 8-bit value except NUL, CR, or LF. However, the extensions will occur within a structured line that uses the DQUOTE, SEMI, SWS, and SP ABNF constructs as delimiters; thus, those delimiter characters MUST be escaped if they would occur within an extension-att-name or extension-att-value. The escape mechanism that MUST be used is the Percent-Encoding defined in Section 2.1 of [RFC3986]. This mechanism is selected as it needs to be supported in an RTSP implementation to deal with URIs anyway. The byte values (in hex) that MUST be escaped are the following: 0x09, 0x20, 0x22, 0x25, and 0x3B.

4.3. ICE Password and Username Transport Header Parameters

The ICE password and username for each agent need to be transported using RTSP. For that purpose, new Transport header parameters are defined (see Section 18.54 of [RFC7826].

There MUST be an "ICE-Password" and "ICE-ufrag" parameter for each media stream. The ICE-ufrag and ICE-Password parameter values MUST be chosen randomly at the beginning of a session. The ICE-ufrag value MUST contain at least 24 bits of randomness, and the ICE-Password value MUST contain at least 128 bits of randomness. This means that the ICE-ufrag value will be at least 4 characters long, and the ICE-Password value at least 22 characters long, since the grammar for these attributes allows for 6 bits of randomness per character. The values 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 is:

```
trns-parameter   =/ SEMI ice-password-par
trns-parameter   =/ SEMI ice-ufrag-par
ice-password-par = "ICE-Password" EQUAL DQUOTE password DQUOTE
ice-ufrag-par    = "ICE-ufrag" EQUAL DQUOTE ufrag DQUOTE
password         = <Defined in [RFC5245], Section 15.4>
ufrag            = <Defined in [RFC5245], Section 15.4>
EQUAL            = <Defined in [RFC7826]>
SEMI             = <Defined in [RFC7826]>
DQUOTE           = <Defined in [RFC7826]>
```

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. Note, this is not a "MUST" as an RTSP client can always attempt to perform a SETUP using ICE to see if it functions or fails. However, including the feature tag in the Supported header ensures that proxies supporting this specification explicitly indicate such support; see Section 7.

4.5. Status Codes

For ICE, there are two new RTSP response codes to indicate progress and errors.

Code	Description	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 Server still working on ICE connectivity checks

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. A client can expect network delays between the server and client resulting in a response longer than 200 milliseconds. Subsequently, every 3 seconds after the previous one was sent, 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 Connectivity check failure

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 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 in response to a PLAY request to indicate 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 an 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 a session-level-only attribute; see [RFC4566].

The ABNF [RFC5234] for the "rtsp-ice-d-m" attribute is:

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

5. ICE-RTSP

This section discusses differences between the regular ICE usage defined in [RFC5245] and ICE-RTSP. The reasons for the differences relate to the clearer client/server roles that RTSP provides and how the RTSP session establishment signaling occurs within RTSP compared to SIP/SDP offer/answer.

5.1. ICE Features Not Required

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

5.1.1. 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 is used to protect against making the RTSP server a denial-of-service tool.

5.1.2. 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.

5.1.3. ICE Remote Candidate Transport Header Parameter

The Remote candidate attribute is not needed for RTSP for the following reasons. Each SETUP request results in an independent ICE processing chain that either fails or results in nominating a single candidate pair to use. If a new SETUP request for the same media is sent, it needs to use a new username fragment and password to avoid any race conditions or uncertainty about to which round of processing the STUN requests relate.

5.2. High-Reachability Configuration

ICE-RTSP contains a high-reachability configuration when the RTSP servers are not behind NATs. Please note that "not behind NATs" may apply in some special cases also for RTSP servers behind NATs given that they are in an address space that has reachability for all the RTSP clients intended to able to reach the server. The high-reachability configuration is similar to ICE-Lite as it allows for some reduction in the server's burden. However, due to the need to still verify that the client is actually present and wants to receive the media stream, the server must also initiate binding requests and await binding responses. The reduction for the high-reachability configuration of ICE-RTSP is that they don't need to initiate their own checks and instead rely on triggered checks for verification. This also removes a denial-of-service threat where an RTSP SETUP request will trigger large amount of STUN connectivity checks towards provided candidate addresses.

6. Detailed Solution

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

6.1. Session Description and RTSP DESCRIBE (Optional)

The RTSP server is RECOMMENDED to 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 exchanges with servers that support ICE thereby 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 the 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". Upon seeing the Supported header, any proxy 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 whether or not all involved parties support ICE. Note that even if a proxy is present in the chain that doesn't indicate support for ICE, it may still work (see Section 7).

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
```

6.2. 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 need to be set up. 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.

6.3. 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 MUST 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 be provided as a fallback in case of proxies that do not support ICE. 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
```

6.4. Gathering Candidates

Upon receiving a SETUP request, the server can determine what media resource should be delivered and which transport alternatives 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 indicate which media protocol is to be used and, based on this and the client's 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 client 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 the server knows should be reachable by a client behind one or more NATs. The reason for this special configuration is twofold: 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. However, note that using this option on a server that doesn't fulfill the requirement of being reachable is counterproductive, 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 S-O connection types to match the client's candidates.

6.5. RTSP Server Response

The server determines if the SETUP request is successful and, if so, returns a 200 OK response; otherwise, it returns 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 6.4 in the "candidates" transport header parameter as well as the server's ICE username fragment and password. In the case that there are no valid candidate pairs with the combination of the client and server candidates, a 480 (ICE Connectivity check failure) error response SHALL be returned, which MUST include the server's candidates. The return of a 480 error may allow both the server and client to release their candidates; see Section 6.10.

Below is an example of a successful response to the request in Section 6.3.

```
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
```

6.6. Server-to-Client ICE Connectivity Checks

The server SHALL start the connectivity checks following the procedures described in Sections 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 server's checks are triggered by the client's connectivity checks.

Please note that Section 5.8 of ICE [RFC5245] does specify that the initiation of the checks are paced and new ones are only started every Ta 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 Ta and RTO 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.

6.7. Client-to-Server ICE Connectivity Check

The client receives the SETUP response and learns the candidate addresses to use for the connectivity checks. The client SHALL initiate its connectivity check(s), following the procedures in Section 6 of ICE [RFC5245]. The pacing of STUN transactions (Appendix B.1 of [RFC5245]) SHALL be used across all media streams that are part of the same RTSP session.

Aggressive nomination SHOULD be used with RTSP during initial SETUP for a resource. This doesn't have all the negative impact that it has in offer/answer as media playing only starts after issuing a PLAY request. Thus, the issue with a change of the media path being used for delivery can be avoided by not issuing a PLAY request while STUN connectivity checks are still outstanding. Aggressive nomination can result in multiple candidate pairs having their nominated flag set, but according to Section 8.1.1.2 of ICE [RFC5245], when the PLAY request is sent, the media will arrive on the pair with the highest priority. Note, different media resources may still end up with different foundations.

The above does not change ICE and its handling of aggressive nomination. When using aggressive nomination, a higher-priority candidate pair with an outstanding connectivity check message can move into the Succeeded state and the candidate pair will have its Nominated flag set. This results in the higher-priority candidate pair being used instead of the previous pair, which is also in the Succeeded state.

To avoid this occurring during actual media transport, the RTSP client can add additional logic when the ICE processing overall is completed to indicate if there are still higher-priority connectivity checks outstanding. If some check is still outstanding, the implementation can choose to wait until some additional timeout is triggered or the outstanding checks complete before progressing with a PLAY request. An alternative is to accept the risk for a path change during media delivery and start playing immediately.

RTSP clients that want to ensure that each media resource uses the same path can use regular nomination where both 1) the ICE processing completion criteria and 2) which media streams are nominated for use can be controlled. This does not affect the RTSP server, as its role is the one of being controlled.

6.8. Client Connectivity Checks Complete

When the client has concluded all of its connectivity checks and has nominated its desired candidate pair 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. The candidate pair with the highest priority will be used for the media. 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 Connectivity check failure) error response. Upon receiving the 480 (ICE Connectivity check failure) 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 (Server still working on ICE connectivity checks) response to indicate this.

6.9. 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 completed successfully. If the server has not concluded its connectivity checks, the server indicates that by sending the 150 (Server still working on ICE connectivity checks) (Section 4.5.1). If there is a problem with the checks, then the server sends 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.

6.10. Freeing Candidates

Both server and client MAY free their non-selected candidates as soon as a 200 OK response has been issued/received for the PLAY request and no outstanding connectivity checks exist.

Clients and servers MAY free all their gathered candidates after having received or sent, respectively, a 480 response to a SETUP request. Clients will likely free their candidates first after having tried any additional actions that may resolve the issue, e.g., verifying the address gathering, or use additional STUN or TURN

servers. Thus, a server will have to weigh the cost of doing address gathering versus maintaining the gathered address for some time to allow any new SETUP request to be issued by the client.

If the 480 response is sent in response to a PLAY request, the server MUST NOT free its gathered candidates. Instead, it will have to wait for additional actions from the client or terminate the RTSP session due to inactivity.

6.11. Steady State

The client and server SHALL use STUN to send keep-alive messages 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.

6.12. Re-SETUP

A client that decides to change any parameters related to the media stream setup will send a new SETUP request. In this new SETUP request, the client MAY include a new different ICE username fragment and password to use in the ICE processing. The new ICE username and password SHALL cause the ICE processing to start from the beginning again, i.e., an ICE restart (Section 9.1.1.1 of [RFC5245]). The client SHALL in case of ICE restart, gather candidates and include the candidates in the transport specification for D-ICE.

ICE restarts may be triggered due to changes of client or server attachment to the network, such as changes to the media streams destination or source address or port. Most RTSP parameter changes would not require an ICE restart, but would use existing mechanisms in RTSP to indicate from what point in the RTP stream they apply. These include the following: performing a pause prior to the parameter change and then resume; assuming the server supports using SETUP during the PLAY state; or using the RTP-Info header (Section 18.45 of [RFC7826]) to indicate from where in the media stream the change shall apply.

Even if the server does not normally support SETUP during PLAY state, it SHALL support SETUP requests in PLAY state for the purpose of changing only the ICE parameters, which are ICE-Password, ICE-ufrag, and the content of ICE candidates.

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 media stream. Once the nomination of the new candidate pair has completed, all unused candidates may be released. If the ICE processing fails and no new candidate pairs are nominated for use, then the media stream MAY continue to use the previously nominated candidate pairs while they still function. If they appear to fail to transport media packets anymore, then the client can select between two actions: attempting any actions that might make ICE work or terminating the RTSP session. Firstly, it can attempt any actions available that might make ICE work, like trying another STUN/TURN server or changing the transport parameters. In that case, the client modifies the RTSP session, and if ICE is still to be used, the client restarts ICE once more. Secondly, if the client is unable to modify the transport or ICE parameters, it MUST NOT restart the ICE processing, and it SHOULD terminate the RTSP session.

6.13. 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. In that case, the server SHALL use the PLAY_NOTIFY method to inform the client (Section 13.5 [RFC7826]) 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 an ice-restart reason, the client SHALL gather new ICE candidates and 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 for each media stream keep-alive messages on the previous set of candidate pairs SHOULD continue until 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

stream, 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
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
      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
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
      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
          Supported: setup.ice-d-m, setup.rtp.rtcp.mux

   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
          Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

7. ICE and Proxies

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

7.1. 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: firstly, a media transport between the server and the proxy according to the proxy's need, and, secondly, delivery from the proxy to the client. This split means that the NAT traversal solution will 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 to support performing NAT traversal (to the server) for clients not capable of NAT traversal present in the same address domain as the proxy.

For a proxy to support the media NAT traversal defined in this specification, a proxy will need to implement the solution fully and be able to act 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.

7.2. Signaling-Only Proxies

A signaling-only proxy handles only the RTSP signaling 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, as discussed in Section 21.2.1 of RTSP 2.0 [RFC7826], usually prevent media delivery to addresses other than from where the RTSP signaling arrives at the server.

The solution for the signaling-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 SHALL indicate its capability by always including the "setup.ice-d-m" feature tag in the Proxy-Supported header in any SETUP request or response.

7.3. 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 since a NAT between the proxy and the client may 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 RTSP 2.0 specification [RFC7826].

The usage of the "setup.ice-d-m" feature tag in the Proxy-Require header is NOT RECOMMENDED because it can have contradictory results. For a proxy that does not support ICE but is media handling, the inclusion of the feature tag will result in aborting the setup and indicating that it isn't supported, which is desirable if providing other fallbacks or other transport configurations to handle the situation is wanted. For non-ICE-supporting non-media-handling proxies, the result will be aborting the setup. However, the setup might have worked if the feature tag wasn't present in the Proxy-Require header. 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 for the intersection of what the proxy chain supports in the Proxy-Supported header. This will provide a positive indication when all proxies in the chain between the client and server support the functionality.

If a proxy doesn't support the "setup.ice-d-m" feature, but that proxy is not a media-handling proxy, the ICE-based setup could still work, since such a proxy may do pass through on any transport parameters. Unfortunately ,the Proxy-Require and Proxy-Supported RTSP headers failed to provide that information. The only way of finding whether or not this is the case is to try perform a SETUP including a Transport header with transport specifications using ICE.

8. 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 is beneficial when combined with ICE for RTSP 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 to negotiate RTP and RTCP multiplexing, see Appendix C of RTSP 2.0 [RFC7826].

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 associated 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 demultiplexer 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 benefits mentioned above, RTSP servers and clients that support "D-ICE" lower-layer transport in combination with RTP SHALL also implement and use RTP and RTCP multiplexing as specified in Appendix C.1.6.4 of [RFC7826] and [RFC5761].

9. Fallback and Using Partial ICE Functionality to Improve NAT/Firewall Traversal

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 to enable interworking with non-ICE capable endpoints. In RTSP, capability determination for ICE can happen prior to the RTSP SETUP request. This means a client should normally not need to include fallback alternatives when offering ICE, as the capability for ICE will already be determined. However, as described in this section, clients may wish to use part of the ICE functionality to improve NAT/firewall traversal where the server is not ICE capable.

Section 4.1.4 of the ICE [RFC5245] specification does recommend that the default destination, i.e., what is used as fallback if the peer isn't ICE capable, is a candidate of relayed type to maximize the likelihood of successful transport of media. This is based on the peer in SIP using SDP offer/answer is almost as likely as the RTSP client to be behind a NAT. For RTSP, the deployment of servers is much more heavily weighted towards deployment with public reachability. In fact, since publicly reachable servers behind NAT either need 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 behind a NAT needs to be configured with a STUN server address to be able to gather candidates successfully, this can be used to derive a server reflexive candidate for the client's port. How useful this is for a NATed RTSP client as a default candidate depends on the properties of the NAT. As long as the NAT uses an address-independent mapping, then using a STUN-derived reflexive candidate is likely to be successful. However, this is brittle in several ways, and the main reason why the original specification of STUN [RFC3489] and direct usage for NAT traversal was obsoleted. First, if the NAT's behavior is attempted to be determined using STUN as described in [RFC3489], the determined behavior 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 includes both address and ports in its response to the SETUP request. Thirdly, messages need 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.

To maximize the likelihood that an RTSP client is capable of receiving media, a relay-based address should be chosen as the default fallback address. However, for RTSP clients lacking a relay server, such as a TURN server, or where usage of such a server has significant cost associated with it, the usage of a STUN-derived server reflexive address as client default has a reasonable likelihood of functioning and may be used as an alternative.

Fallback addresses need to be provided in their own transport specification using a specifier that does 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.

10. IANA Considerations

Per this document, registrations have been made in a number of registries, both for RTSP and SDP. For all the below registrations, the contact person on behalf of the IETF WG MMUSIC is Magnus Westerlund <magnus.westerlund@ericsson.com>.

10.1. RTSP Feature Tags

Per this document, one RTSP 2.0 feature tag has been registered in the "RTSP 2.0 Feature-tags" registry.

setup.ice-d-m: A feature tag representing the support of the ICE-based establishment of datagram media transport that is capable of transport establishment through NAT and firewalls. This feature tag applies to clients, servers, and proxies and indicates support of all the mandatory functions of this specification.

10.2. Transport Protocol Identifiers

Per this document, a number of transport protocol combinations have been registered in the RTSP 2.0 "Transport Protocol Identifiers" 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.

10.3. RTSP Transport Parameters

Per this document, three transport parameters have been registered in the RTSP 2.0's "Transport Parameters" registry.

candidates: Listing the properties of one or more ICE candidates. See Section 4.2.

ICE-Password: The ICE password used to authenticate the STUN binding request in the ICE connectivity checks. See Section 4.3.

ICE-ufrag: The ICE username fragment used to authenticate the STUN binding requests in the ICE connectivity checks. See Section 4.3.

10.4. RTSP Status Codes

Per this document, two assignments have been made in the "RTSP 2.0 Status Codes" registry. See Section 4.5.

10.5. Notify-Reason Value

Per this document, one assignment has been made in the RTSP 2.0 Notify-Reason header value registry. The defined value is:

ice-restart: This Notify-Reason value allows the server to notify the client about the need for an ICE restart. See Section 4.6.

10.6. SDP Attribute

One SDP attribute has been registered:

SDP Attribute ("att-field"):

Attribute name:	rtsp-ice-d-m
Long form:	ICE for RTSP datagram media NAT traversal
Type of attribute:	Session-level only
Subject to charset:	No
Purpose:	RFC 7825, Section 4.7
Values:	No values defined
Contact:	Magnus Westerlund Email: magnus.westerlund@ericsson.com Phone: +46 10 714 82 87

11. Security Considerations

ICE [RFC5245] and ICE TCP [RFC6544] provide an extensive discussion on security considerations that apply here as well.

11.1. 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 that 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 play-out 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 would otherwise be only signaling proxies.

To protect against attacks on ICE based on signaling information, RTSP signaling SHOULD be protected using TLS to prevent eavesdropping and modification of information.

The STUN amplification attack described in Section 18.5.2 in ICE [RFC5245] needs consideration. Servers that are able to run according to the high-reachability option have good mitigation of this attack as they only send connectivity checks towards an address and port pair from which they have received an incoming connectivity check. This means an attacker requires both the capability to spoof source addresses and 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.

11.2. Logging

The logging of NAT translations is helpful to analysts, particularly in enterprises, who need to be able to map sessions when investigating possible issues where the NAT happens. When using logging on the public Internet, it is possible that the logs are large and privacy invasive, so procedures for log flushing and

privacy protection SHALL be in place. Care should be taken in the protection of these logs and consideration taken to log integrity, privacy protection, and purging logs (retention policies, etc.). Also, logging of connection errors and other messages established by this document can be important.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <http://www.rfc-editor.org/info/rfc5761>.

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, DOI 10.17487/RFC6544, March 2012, <http://www.rfc-editor.org/info/rfc6544>.

[RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <http://www.rfc-editor.org/info/rfc7826>.

12.2. Informative References

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, DOI 10.17487/RFC2326, April 1998, <http://www.rfc-editor.org/info/rfc2326>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <http://www.rfc-editor.org/info/rfc3022>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

[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, DOI 10.17487/RFC3489, March 2003, <http://www.rfc-editor.org/info/rfc3489>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <http://www.rfc-editor.org/info/rfc4340>.

[RFC7604] Westerlund, M. and T. Zeng, "Comparison of Different NAT Traversal Techniques for Media Controlled by the Real-Time Streaming Protocol (RTSP)", RFC 7604, DOI 10.17487/RFC7604, September 2015, <http://www.rfc-editor.org/info/rfc7604>.

Acknowledgments

The authors would like to thank: Remi Denis-Courmont for suggesting the method of integrating ICE in RTSP signaling, Dan Wing for help with the security section and numerous other issues, Ari Keranen for review of the document and its ICE details, and Flemming Andreasen and Alissa Cooper for a thorough review. In addition, Bill Atwood has provided comments and suggestions for improvements.

Authors' Addresses

Jeff Goldberg
Cisco
32 Hamelacha St.
South Netanya 42504
Israel

Phone: +972 9 8927222
Email: jgoldber@cisco.com

Magnus Westerlund
Ericsson
Farogatan 6
Stockholm SE-164 80
Sweden

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

Thomas Zeng
Nextwave Wireless, Inc.
12670 High Bluff Drive
San Diego, CA 92130
United States of America

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