Mapping ICE (Interactive Connectivity Establishment) to RTSP

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on Aug 8, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This memo describes a mapping from ICE (Interactive Connectivity Establishment) to RTSP for the purpose of Network Address Translator (NAT) traversal for RTSP protocol. In order to become compatible with ICE, the Transport header in RTSP is extended with new syntax elements. This memo presents a few examples RTSP conversations that uses ICE for NAT/firewall traversals.
1. Introduction

ICE protocol is a proposed framework for NAT and firewall traversal.

In [1], the parameters for various ICE messages are defined in generic XML syntax. Each multimedia signalling protocol needs to map these parameters to its own protocol parameters. Section 9 of [1] provides a mapping for SIP (Session Initiation Protocol) based on the SDP Offer/Answer model.

This memo provides a mapping for RTSP (Real-Time Streaming Protocol).

Unlike SIP, RTSP is a multimedia signalling protocol that does not follow the SDP Offer/Answer model defined in RFC3264, for historical reasons. It is therefore necessary to extend the Transport header in RTSP with new syntax elements in order to fully implement ICE features.

The readers of this memo are expected to have read [1] (especially sections 5 and 9) and have gained a reasonable understand of ICE framework.

RTSP differs from SIP in that RTSP server and RTSP client are almost never deployed behind different NAT/firewalls at the same time. That is, either RTSP server or RTSP client is in the open.

The examples in this memo limit the traversal problem to

1) RTSP server in the open;
2) RTSP client in the open.

In such cases, TURN services are not required for connectivity establishment between RTSP server and client.
2. Mapping ICE to RTSP

2.1 How Does ICE Work For RTSP: an overview

The key assumption in ICE is that a signalling entity cannot know, apriori, whether the peer it wishes to communicate with is connected to one or all of the address realms it is in. Therefore, in order to communicate, it has to try them all, and choose the best one that works. This assumption is true for RTSP.

As described in figure 1, section 2 of [1], in terms of signalling model for RTSP, the initiator is the RTSP client, the responder is the RTSP server, the initiate message is a SETUP message, and the accept message is a SETUP response. The modify message is a SETUP message, and the modify acceptance message is a SETUP response.

It is also an option to treat the DESCRIBE response from RTSP server to RTSP client as another initiate message in the ICE context. For RTSP, DESCRIBE response normally carries the session description in SDP format. It is in this SDP that the RTSP server may use the SDP extension in section 9 of [1] to inform its client of the addresses, ports and associated parameters (e.g., user name, password) that the server has discovered. However, since not all RTSP sessions begin with DESCRIBE (many rely on ftp or HTTP protocols to obtain session descriptions out of band), in the rest of this memo, we will only consider SETUP as the initiate message, even though starting ICE process with DESCRIBE response can save up to one round of ICE negotiations.

Here is how ICE would work with RTSP.

Before the RTSP client establishes a session, it obtains as many IP address and port combinations in as many address realms as it can. Any protocol that provides a client with an IP address and port on which the RTSP client can receive traffic can be used. These include STUN and even VPN. The RTSP client also uses any local interface addresses. A dual-stack v4/v6 client will obtain both a v6 and a v4 address/port. The only requirement is that, across all of these addresses, the RTSP client can be certain that at least one of them will work for any responder it might communicate with. This is guaranteed by:

1) The assumption that the RTSP client and server are separated by at most one level of NAT/firewall;
2) The assumption that co-located STUN servers can be installed on the media ports in each protocol entity.

The RTSP client then makes a STUN server available on each of the address/port combinations it has obtained. This STUN server is running locally, on the initiator. All of these addresses are placed into the Tranport header of the SETUP request and they are ordered in terms of preference given in [1]. The SETUP request also conveys the STUN username and password which are required to gain access to the STUN server on each address/port combination. Tranport header extensions are described in the next section to convey username and
password.

The initiate message -- the SETUP request, is sent to the responder (normally RTSP server) via the RTSP connection, preferably using a secure protocol such as TLS.

Once the RTSP server receives the SETUP request, it sends STUN requests to each alternate address/
port in the Transport. These STUN requests include the username and password obtained from the initiate message. The STUN requests serve two purposes. The first is to check for connectivity. If a response is received, the RTSP server knows that it can reach the client at that address. The second purpose is to obtain more addresses at which the RTSP server can be contacted. If the client is behind a NAT, the RTSP server may discover another address through the STUN responses. In its accept message -- 200 OK Setup response, the RTSP server includes all addresses that it can unilaterally determine (just as the client did), in addition to any that were discovered using the STUN messages to the RTSP client.

When the accept message arrives at the RTSP client, the client performs a similar operation. Using STUN, it checks connectivity to each of the addresses in the accept message. Through the STUN responses, it may learn of additional addresses that it can use to receive media. If it does learn any new address, the client generates a modify message to pass this address to the RTSP server. For RTSP, modify message is re-SETUP request. The RTSP server processes the re-SETUP request as a "ICE Modify" message and sends a "200 OK" SETUP response as the "ICE Modify response" message.

At this point, ICE process is complete, or else connection cannot be established.

2.1 Extending RTSP Transport Header Syntax

In order to convey username and password used to access colocated STUN servers, it is necessary to extend the RTSP Transport header definitions in [4].

Transport = "Transport" ":" 1#transport-spec
transport-spec = transport-id *parameter
transport-id = transport-protocol "/" profile
["/" lower-transport]
; no LWS is allowed inside transport-id
transport-protocol = "RTP" / token
profile = "AVP" / token
lower-transport = "TCP" / "UDP" / token
parameter = ";" ( "unicast" / "multicast" )
...
3. Terminology

Several new terms are introduced in [1] and elaborated in this memo:

Session Initiator: A software entity that, at the request of a user, tries to establish communications with another entity, called the session responder. A session initiator is also called an initiator. In RTSP context, initiator is normally the RTSP client.

Initiator: Another term for a session initiator.

Session Responder: A software entity that receives a request for establishment of communications from the session initiator, and either accepts or declines the request. A session responder is also called a responder. In RTSP context, a session responder is normally the RTSP server.

Responder: Another term for a session responder.

Initiate Message: The signaling message used by an initiator to establish communications. It contains capabilities and other information needed by the responder to send media to the initiator.

Accept Message: The signaling message used by a responder to agree to communications. It contains capabilities and other information needed by the initiator to send media to the responder.

Modify Message: The signaling message used by either an initiator or responder to change the capability and other information needed by the peer for sending media.

Modify Acceptance Message: The signaling message used by a client to agree to the changes proposed in a modify message, and to present the capability or other information needed by its peer for sending media.

Protocol Entity: either side of the media stream. For RTSP, a protocol entity is either the RTSP server or the RTSP client.
Terminate Message The signaling message used by a client to terminate the session and associated media streams.

Transport Address: The combination of an IP address and port.

Local Transport Address: A local transport address is transport address that has been allocated from the operating system on the host. This includes transport addresses obtained through VPNs, and also transport addresses obtained through RSIP (which lives at the operating system level). Transport addresses are typically obtained by binding to an interface.

Derived Transport Address: A derived transport address is a transport address which is associated with, but different from, a local transport address. The derived transport address is associated with the local transport address in that packets sent to the derived transport address are received on the socket bound to that local transport address. Derived addresses are obtained using protocols like STUN and TURN, and more generally, any UNSAF protocol [11].
4. Example RTSP Conversations

The examples below follow the RTSP ABNF rules in [4]. It is worth noting that some of the syntax elements, such as "dest_addr" and "src_addr", are new to [4], and were not in RFC2326.

4.1 Case 1: RTSP server is in the open; client behind cone NAT

The following sample RTSP conversation describes how ICE traverses a "cone" NAT on behalf of an RTSP client behind NAT. In this example, RTSP server is in the public address realm, which is true for most RTSP service deployment to-date.

Recall from [5] the definition of cone NATs:

- **Full Cone**: A full cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Furthermore, any external host can send a packet to the internal host, by sending a packet to the mapped external address.

- **Restricted Cone**: A restricted cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Unlike a full cone NAT, an external host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X.

We assume that the RTSP client behind this cone NAT obtains its external IP (i.e., 24.2.1.1) and port apriori, using a public STUN server. The RTSP client’s first SETUP request includes two choices of addresses for RTP/RTCP ports, as shown by the relevant RTSP conversation below:

```
C->S  SETUP rtp://foo.com/test.wav/streamid=0 RTSP/1.0
     Transport: RTP/AVP/UDP;unicast;src_addr="172.16.1.1:6970"/
                   "172.16.1.1:6971"; username="foo"; password="x",
                   RTP/AVP/UDP;unicast;src_addr="24.2.1.1:9970"/
                   "24.2.1.1:9980"; username="server"; password="s"
     CSeq: 2
S->C  RTSP/1.0 200 OK
     Transport: RTP/AVP/UDP;unicast;dest_addr="24.2.1.1:9970"/
                   "24.2.1.1:9980"; src_addr="24.2.8.8:5540"/
                   "24.2.8.8:5541"; username="client"; password="c"
     CSeq: 2
     Session: 2034820394
```

Comments: in the first SETUP request message, there are two transport specifications, separated by a comma as per [4]. The first transport uses local address and port, while the second uses STUN discovered public address and port. In the 200 OK response, the presence of "dest_addr" parameter indicates that RTSP server has completed its ICE process after successful STUN bindings.

Finally RTSP client performs STUN bindings against the RTSP server...
using the "src_addr", username and password in the SETUP request, and receives STUN responses.

In this example, connectivity is established in only one round of ICE negotiation, thanks to the fact that STUN binding is performed appropriately. A nice benefit is that RTSP conversational delay is not increased by much. But connectivity may not always be established in one SETUP / Response cycle. In the case of symmetric NAT, STUN binding must be done during RTSP conversations, not before, as shown by the next example.

4.2 Case 2: RTSP server is in the open; client behind Symmetric NAT

In this case, obtaining external IP address and port apriori is of no value, given the symmetric nature of the NAT. Therefore, the RTSP client does not list any public address in its first SETUP request.

C->S  SETUP rtsp://foo.com/test.wav/streamid=0 RTSP/1.0
Transport: RTP/AVP/UDP;unicast;src_addr="172.16.1.1:6970"/
"172.16.1.1:6971"; username="foo"; password="x"
CSeq: 2

/* RTSP server cannot reach "172.16.1.1". The server’s STUN binding request will timeout, and it then sends the following response. The lesson here is that STUN binding timeout should be set to a fairly short value so as to minimize the impact on RTSP delay. */

S->C  RTSP/1.0 200 OK
Transport: RTP/AVP/UDP;unicast; src_addr="24.2.8.8:5540"/
"24.2.8.8:5541"; username="client"; password="c"
CSeq: 2
Session: 2034820394

/* RTSP client now performs STUN bindings and finds its external address/port pair as, say, "24.2.1.1:6970"/"24.2.1.1:6971", it then sends re-SETUP as ICE modify message: */

C->S  SETUP rtsp://foo.com/test.wav/streamid=0 RTSP/1.0
Transport: RTP/AVP/UDP;unicast;src_addr="172.16.1.1:6970"/
"172.16.1.1:6971"; username="foo"; password="x"
RTP/AVP/UDP;unicast;src_addr="24.2.1.1:6970"/
"24.2.1.1:6971"; username="server"; password="s"
CSeq: 3
Session: 2034820394

/* RTSP server can reach 24.2.1.1. So it sends the following 200 OK: */

S->C  RTSP/1.0 200 OK
Transport: RTP/AVP/UDP;unicast;dest_addr="24.2.1.1:6970"/
"24.2.1.1:6971"; src_addr="24.2.8.8:5540"/
"24.2.8.8:5541"; username="client"; password="c"
CSeq: 3
Session: 2034820394

Comments: RTSP SETUP delay has been increased in this case by two factors, when compared to case 1:

1) STUN timeout after the first SETUP request is received by RTSP
server.
2) Additional SETUP/response round trip.

Case 3: RTSP client is in the open, RTSP server is behind symmetric NAT

In this scenario, client has only one address to include in its SETUP request.

C->S SETUP rtsp://foo.com/test.wav/streamid=0 RTSP/1.0
    Transport: RTP/AVP/UDP;unicast;src_addr="24.2.1.1:6970"/
    "24.2.1.1:6971"; username="server";
    password="x"
    CSeq: 2

S->C RTSP/1.0 200 OK
    Transport: RTP/AVP/UDP;unicast; dest_addr="24.2.8.8:5540"/
    "24.2.8.8:5541"; src_addr="24.2.8.8:5540"/
    "24.2.8.8:5541"; username="client"; password="c"
    CSeq: 2
    Session: 2034820394

Here no additional RTSP message exchange is needed.

5. Security Considerations

The sections titled "security considerations" in [1] and [4] covers all the security considerations relevant to this memo. No additional consideration is deemed necessary.
Normative References


Author’s Address

Thomas Zeng
PV Network Solutions,
10350 Science Center Dr., Suite 200
San Diego, CA92127
US

Phone: +1 858 731 5465
EMail: zeng@pv.com
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF’s procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the
Internet Society.