Network Working Group	M. Westerlund
Internet-Draft	Ericsson
Intended status: Informational	T. Zeng
Expires: April 29, 2012	October 27, 2011

The Evaluation of Different Network Addres Translator (NAT) Traversal Techniques for Media Controlled by Real-time Streaming Protocol (RTSP)
draft-ietf-mmusic-rtsp-nat-evaluation-04

Abstract

This document describes several Network Address Translator (NAT) traversal techniques that was considered to be used by Real-time Streaming Protocol (RTSP). Each technique includes a description on how it would be used, the security implications of using it and any other deployment considerations it has. There are also disussions on how NAT traversal techniques relates to firewalls and how each technique can be applied in different use cases. These findings where used when selecting the NAT traversal for RTSP 2.0 standardized in the MMUSIC WG.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Today there is a proliferate deployment of different flavors of Network Address Translator (NAT) boxes that in many cases only loosely follows standards *[RFC3022]*[RFC2663][RFC3424]. NATs cause discontinuity in address realms *[RFC3424]*, therefore an application protocol, such as Real-time Streaming Protocol (RTSP) *[RFC2326]*[I-D.ietf-mmusic-rfc2326bis], needs to deal with such discontinuities caused by NATs. The problem is that, being a media control protocol managing one or more media streams, RTSP carries network address and port information within its protocol messages. Because of this, even if RTSP itself, when carried over Transmission Control Protocol (TCP) *[RFC0793]* for example, may not be blocked by NATs, its media streams may be blocked by NATs. This will occur unless special protocol provisions are added to support NAT-traversal.

Like NATs, firewalls (FWs) are also middle boxes that need to be considered. Firewalls helps prevent unwanted traffic from getting in or out of the protected network. RTSP is designed such that a firewall can be configured to let RTSP controlled media streams to go through with minimal implementation effort. The minimal effort is to implement an Application Level Gateway (ALG) to interpret RTSP parameters. There is also a large class of firewalls, commonly home firewalls, that uses a similar filtering behavior to what NAT has. This type of firewalls can be handled using the same solution as employed for NAT traversal instead of relying on ALGs.

This document describes several NAT-traversal mechanisms for RTSP controlled media streaming. These NAT solutions fall into the category of "UNilateral Self-Address Fixing (UNSAF)" as defined in [RFC3424] and quoted below:

"UNSAF is a process whereby some originating process attempts to determine or fix the address (and port) by which it is known - e.g. to be able to use address data in the protocol exchange, or to advertise a public address from which it will receive connections."

Following the guidelines spelled out in RFC 3424, we describe the required RTSP protocol extensions for each method, transition strategies, and security concerns.

This document is capturing the evaluation done in the process to recommend FW/NAT traversal methods for RTSP streaming servers based on RFC 2326 *[RFC2326]* as well as the RTSP 2.0 core spec *[I-D.ietf-mmusic-rfc2326bis]*. The evaluation is focused on NAT traversal for the media streams carried over User Datagram Protocol (UDP) *[RFC0768]*. Where Real-time Transport Protocol (RTP) *[RFC3550]* over UDP being the main

case for such usage. The findings should be applicable to other protocols as long as they have similar properties.

1.1. Network Address Translators

Readers are urged to refer to "IP Network Address Translator (NAT) Terminology and Considerations" *[RFC2663]* for information on NAT taxonomy and terminology. Traditional NAT is the most common type of NAT device deployed. Readers may refer to "Traditional IP Network Address Translator (Traditional NAT)" *[RFC3022]* for detailed information on traditional NAT. Traditional NAT has two main varieties -- Basic NAT and Network Address/Port Translator (NAPT).
NAPT is by far the most commonly deployed NAT device. NAPT allows multiple internal hosts to share a single public IP address simultaneously. When an internal host opens an outgoing TCP or UDP session through a NAPT, the NAPT assigns the session a public IP address and port number, so that subsequent response packets from the external endpoint can be received by the NAPT, translated, and forwarded to the internal host. The effect is that the NAPT establishes a NAT mapping to translate the (private IP address, private port number) tuple to a (public IP address, public port number) tuple, and vice versa, for the duration of the session. An issue of relevance to peer-to-peer applications is how the NAT behaves when an internal host initiates multiple simultaneous sessions from a single (private IP, private port) endpoint to multiple distinct endpoints on the external network. In this specification, the term "NAT" refers to both "Basic NAT" and "Network Address/Port Translator (NAPT)".
This document uses the term "address and port mapping" as the translation between an external address and port and an internal address and port. Note that this is not the same as an "address binding" as defined in RFC 2663. There exist a number of address and port mapping behaviors described in more detail in Section 4.1 of "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP" *[RFC4787]*.
NATs also have a filtering behavior on traffic arriving on the external side. Such behavior effects how well different methods for NAT traversal works through these NATs. See Section 5 of "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP" *[RFC4787]* for more information on the different types of filtering that have been identified.

1.2. Firewalls

A firewall (FW) is a security gateway that enforces certain access control policies between two network administrative domains: a private domain (intranet) and a external domain, e.g. public Internet. Many organizations use firewalls to prevent privacy intrusions and malicious attacks to corporate computing resources in the private intranet [RFC2588].

A comparison between NAT and FW is given below:

1. A firewall must sit between two network administrative domains, while NAT does not have to sit between two domains.

2. NAT does not in itself provide security, although some access control policies can be implemented using address translation schemes. The inherent filtering behaviours are commonly mistaken for real security policies.

It should be noted that many NAT devices intended for small office/home office (SOHO) include both NATs and firewall functionality.
In the rest of this memo we use the phrase "NAT traversal" interchangeably with "FW traversal", "NAT/FW traversal" and "NAT/ Firewall traversal".

1.3. Glossary

ALG: Application Level Gateway, an entity that can be embedded in a NAT or other middlebox to perform the application layer functions required for a particular protocol to traverse the NAT/middlebox.

ICE: Interactive Connectivity Establishment, see [RFC5245].

DNS: Domain Name Service

DDOS: Distributed Denial Of Service attacks

NAT: Network Address Translator, see [RFC3022].

NAPT: Network Address/Port Translator, see [RFC3022].

RTP: Real-time Transport Protocol, see [RFC3550].

RTSP: Real-Time Streaming Protocol, see [RFC2326] and [I-D.ietf-mmusic-rfc2326bis].

SDP: Session Description Protocol, see [RFC4566].

SSRC: Synchronization source in RTP, see [RFC3550].

1.4. Definitions

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]*.

2. Detecting the loss of NAT mappings

Several NAT traversal techniques in the next chapter make use of the fact that the NAT UDP mapping's external address and port can be discovered. This information is then utilized to traverse the NAT box. However any such information is only good while the mapping is still valid. As the IAB's UNSAF document [RFC3424] points out, the mapping can either timeout or change its properties. It is therefore important for the NAT traversal solutions to handle the loss or change of NAT mappings, according to RFC3424.
First, since NATs may also dynamically reclaim or readjust address/port translations, "keep-alive" and periodic re-polling may be required according to RFC 3424. Secondly, it is possible to detect and recover from the situation where the mapping has been changed or removed. The loss of a mapping can be detected when no traffic arrives for a while. Below we will give some recommendation on how to detect loss of NAT mappings when using RTP/RTCP under RTSP control.
A RTP session normally has both RTP and RTCP streams. The loss of a RTP mapping can only be detected when expected traffic does not arrive. If a client does not receive data within a few seconds after having received the "200 OK" response to a PLAY request, there are likely some middleboxes blocking the traffic. However, for a receiver to be more certain to detect the case where no RTP traffic was delivered due to NAT trouble, one should monitor the RTCP Sender reports. The sender report carries a field telling how many packets the server has sent. If that has increased and no RTP packets has arrived for a few seconds it is likely the RTP mapping has been removed.
The loss of mapping for RTCP is simpler to detect. RTCP is normally sent periodically in each direction, even during the RTSP ready state. If RTCP packets are missing for several RTCP intervals, the mapping is likely to be lost. Note that if neither RTCP packets nor RTSP messages are received by the RTSP server for a while, the RTSP server has the option to delete the corresponding RTP session, SSRC and RTSP session ID, because either the client can not get through a middle box NAT/FW, or that the client is mal-functioning.

3. Requirements on NAT-Traversal

This section considers the set of requirements for the evaulation of RTSP NAT traversal solutions.
RTSP is a client-server protocol. Typically services providers deploy RTSP servers in the public address realm. However, there are use cases where the reverse is true: RTSP clients are connecting from public address realm to RTSP servers behind home NATs. This is the case for instance when home surveillance cameras running as RTSP servers intend to stream video to cell phone users in the public address realm through a home NAT. In terms of requirements, the first requirement should be to solve the RTSP NAT traversal problem for RTSP servers deployed in a public network, i.e. no NAT at the server side.

The list of feature requirements for RTSP NAT solutions are given below:

1. MUST work for all flavors of NATs, including NATs with address and port restricted filtering.

2. MUST work for firewalls (subject to pertinent firewall administrative policies), including those with ALGs.

3. SHOULD have minimal impact on clients in the open and not dual-hosted. RTSP dual-hosting means that RTSP protocol and the media protocol (e.g. RTP) are implemented on different computers with different IP addresses.

   *For instance, no extra delay from RTSP connection till arrival of media

4. SHOULD be simple to use/implement/administer that people actually turn them on

   *Otherwise people will resort to TCP tunneling through NATs

   *Address discovery for NAT traversal should take place behind the scene, if possible

5. SHOULD authenticate dual-hosted client transport handler to prevent DDOS attacks.

The last requirement addresses the Distributed Denial-Of-Service (DDOS) threat, which relates to NAT traversal as explained below.
During NAT traversal, when the RTSP server determines the media destination (Address and port) for client, the result may be that the public IP address of the RTP receiver host is different than the public IP address of the RTSP client host. This posts a DDOS threat that has significant amplification potentials because the RTP media streams in general consist of large number of IP packets. DDOS attacks occur if the attacker fakes the messages in the NAT traversal mechanism to trick the RTSP server into believing that the client's RTP receiver is located in a separate host. For example, user A may use his RTSP client to direct the RTSP server to send video RTP streams to target.example.com in order to degrade the services provided by target.example.com. Note a simple preventative measure is for the RTSP server to disallow the cases where the client's RTP receiver has a different public IP address than that of the RTSP client. However, in some applications (e.g., centralized conferencing), dual-hosted RTSP/RTP clients have valid use cases. The key is how to authenticate the messages exchanged during the NAT traversal process. Message authentication is a big challenge in the current wired and wireless networking environment. It may be necessary in the immediate future to deploy NAT traversal solutions that do not have full message

authentication, but provide upgrade path to add authentication features in the future.

4. NAT Traversal Techniques

There exist a number of potential NAT traversal techniques that can be used to allow RTSP to traverse NATs. They have different features and are applicable to different topologies; their cost is also different. They also vary in security levels. In the following sections, each technique is outlined in details with discussions on the corresponding advantages and disadvantages.
This section includes NAT traversal techniques that have not been formally specified anywhere else. The overview section of this document may be the only publicly available specification of some of the NAT traversal techniques. However that is no real barrier against doing an evaluation of the NAT traversal technique. Some other techniques are currently (at the time of writing) in a state of flux due to ongoing standardization work on these techniques, e.g. RTP No-Op [I-D.ietf-avt-rtp-no-op].

4.1. STUN

4.1.1. Introduction

STUN - "Simple Traversal of UDP Through Network Address Translators" [RFC3489][RFC5389] is a standardized protocol that allows a client to use secure means to discover the presence of a NAT between himself and the STUN server. The client uses the STUN server to discover the address mappings assigned by the NAT. STUN is a client-server protocol. STUN client sends a request to a STUN server and the server returns a response. There are two types of STUN requests - Binding Requests, sent over UDP, and Shared Secret Requests, sent over TLS over TCP.
The first version of STUN *[RFC3489]* included categorization and parameterization of NATs. This was abandoned in the updated version due to it being unreliable.

4.1.2. Using STUN to traverse NAT without server modifications

This section describes how a client can use STUN to traverse NATs to RTSP servers without requiring server modifications. Note that this method has limited applicability and requires the server to be available in the external/public address realm in regards to the client located behind a NAT(s).
Limitations:

*The server must be located in either a public address realm or the next hop external address realm in regards to the client.

*The client may only be located behind NATs that performing Endpoint Independent or Address Dependent Mappings. Clients

behind NATs that do Address and Port Dependent Mappings cannot use this method.

Method:
A RTSP client using RTP transport over UDP can use STUN to traverse a NAT(s) in the following way:

1. Use STUN to try to discover the type of NAT, and the timeout period for any UDP mapping on the NAT. This is RECOMMENDED to be performed in the background as soon as IP connectivity is established. If this is performed prior to establishing a streaming session the delays in the session establishment will be reduced. If no NAT is detected, normal SETUP SHOULD be used.

2. The RTSP client determines the number of UDP ports needed by counting the number of needed media transport protocols sessions in the multi-media presentation. This information is available in the media description protocol, e.g. SDP [RFC4566]. For example, each RTP session will in general require two UDP ports, one for RTP, and one for RTCP.

3. For each UDP port required, establish a mapping and discover the public/external IP address and port number with the help of the STUN server. A successful mapping looks like: client's local address/port <-> public address/port.

4. Perform the RTSP SETUP for each media. In the transport header the following parameter SHOULD be included with the given values: "dest_addr" *[I-D.ietf-mmusic-rfc2326bis]* or "destination" + "client_port" *[RFC2326]* with the public/external IP address and port pair for both RTP and RTCP. To be certain that this works servers must allow a client to setup the RTP stream on any port, not only even ports and with non-continuous port numbers for RTP and RTCP. This requires the new feature provided in the update to RFC2326 *[I-D.ietf-mmusic-rfc2326bis]*. The server should respond with a transport header containing an "src_addr" or "source parameter" + "server_port" with the RTP and RTCP source IP address and port of the media stream.

5. To keep the mappings alive, the client SHOULD periodically send UDP traffic over all mappings needed for the session. For the mapping carrying RTCP traffic the periodic RTCP traffic may be enough. For mappings carrying RTP traffic and for mappings carrying RTCP packets at too low a frequency, keep-alive messages SHOULD be sent. As keep alive messages, one could use the RTP No-Op packet *[I-D.ietf-avt-rtp-no-op]* to the streaming server's discard port (port number 9). The drawback of using RTP No-Op is that the payload type number must be dynamically

assigned through RTSP first. Otherwise STUN could be used for the keep-alive as well as empty UDP packets.

If a UDP mapping is lost, the above discovery process must be repeated. The media stream also needs to be SETUP again to change the transport parameters to the new ones. This will cause a glitch in media playback. To allow UDP packets to arrive from the server to a client behind a "Address Dependent" filtering NAT, the client must first send a UDP packet to establish filtering state in the NAT. The client, before sending a RTSP PLAY request, must send a so called FW packet (such as a RTP No-Op packet) on each mapping, to the IP address given as the servers source address. To create minimum problems for the server these UDP packets SHOULD be sent to the server's discard port (port number 9). Since UDP packets are inherently unreliable, to ensure that at least one UDP message passes the NAT, FW packets should be retransmitted a reasonable number of times.
For a "Address and Port Dependent" filtering NAT the client must send messages to the exact ports used by the server to send UDP packets before sending a RTSP PLAY request. This makes it possible to use the above described process with the following additional restrictions: for each port mapping, FW packets need to be sent first to the server's source address/port. To minimize potential effects on the server from these messages the following type of FW packets MUST be sent. RTP: an empty or less than 12 bytes UDP packet. RTCP: A correctly formatted RTCP RR or SR message. The above described adaptations for restricted NATs will not work unless the server includes the "src_addr" in the "Transport" header (which is the "source" transport parameter in RFC2326).
This method is also brittle because it relies on that one can use STUN to classify the NAT behavior. If the NAT changes the properties of the existing mapping and filtering state for example due to load, then the methods will fail.

4.1.3. Embedding STUN in RTSP

This section outlines the adaptation and embedding of STUN within RTSP. This enables STUN to be used to traverse any type of NAT, including symmetric NATs. This would require protocol changes.
This NAT traversal solution has limitations:

1. It does not work if both RTSP client and RTSP server are behind separate NATs.

2. The RTSP server may, for security reasons, refuse to send media streams to an IP different from the IP in the client RTSP requests.

Deviations from STUN as defined in RFC 3489:

1. We allow RTSP applications to have the option to perform STUN "Shared Secret Request" through RTSP, via extension to RTSP;

2. We require STUN server to be co-located on RTSP server's media output ports.

In order to allow binding discovery without authentication, the STUN server embedded in RTSP application must ignore authentication tag, and the STUN client embedded in RTSP application must use dummy authentication tag.
If STUN server is co-located with RTSP server's media output port, an RTSP client using RTP transport over UDP can use STUN to traverse ALL types of NATs. In the case of port and address dependent mapping NATs, the party inside the NAT must initiate UDP traffic. The STUN Bind Request, being a UDP packet itself, can serve as the traffic initiating packet. Subsequently, both the STUN Binding Response packets and the RTP/RTCP packets can traverse the NAT, regardless of whether the RTSP server or the RTSP client is behind NAT.
Likewise, if an RTSP server is behind a NAT, then an embedded STUN server must co-locate on the RTSP client's RTCP port. Also it will become the client that needs to disclose his destination address rather than the server so that the server correctly can determine its NAT external source address for the media streams. In this case, we assume that the client has some means of establishing TCP connection to the RTSP server behind NAT so as to exchange RTSP messages with the RTSP server.
To minimize delay, we require that the RTSP server supporting this option must inform its client the RTP and RTCP ports from where the server intend to send out RTP and RTCP packets, respectively. This can be done by using the "server_port" parameter in RFC2326, and the "src_addr" parameter in [I-D.ietf-mmusic-rfc2326bis]. Both are in the RTSP Transport header. But in general this strategy will require that one first do one SETUP request per media to learn the server ports, then perform the STUN checks, followed by a subsequent SETUP to change the client port and destination address to what was learned during the STUN checks.
To be certain that RTCP works correctly the RTSP end-point (server or client) will be required to send and receive RTCP packets from the same port.

4.1.4. Discussion On Co-located STUN Server

In order to use STUN to traverse "address and port dependent" filtering or mapping NATs the STUN server needs to be co-located with the streaming server media output ports. This creates a de-multiplexing problem: we must be able to differentiate a STUN packet from a media packet. This will be done based on heuristics. A common heuristics is the first byte in the packet, which works fine between STUN and RTP or

RTCP where the first byte happens to be different, but may not work as well with other media transport protocols.

4.1.5. ALG considerations

If a NAT supports RTSP ALG (Application Level Gateway) and is not aware of the STUN traversal option, service failure may happen, because a client discovers its public IP address and port numbers, and inserts them in its SETUP requests, when the RTSP ALG processes the SETUP request it may change the destination and port number, resulting in unpredictable behavior. An ALG should not update address fields which contains addresses other than the NATs internal address domain. In cases where the ALG modifies fields unnecessary two alternatives exist:

1. The usage of TLS to encrypt the RTSP TCP connection to prevent the ALG from reading and modifying the RTSP messages.

2. To turn off the STUN based NAT traversal mechanism

As it may be difficult to determine why the failure occurs, the usage of TLS protected RTSP message exchange at all times would avoid this issue.

4.1.6. Deployment Considerations

For the non-embedded usage of STUN the following applies:
Advantages:

*STUN is a solution first used by SIP applications. As shown above, with little or no changes, RTSP application can re-use STUN as a NAT traversal solution, avoiding the pit-fall of solving a problem twice.

*Using STUN does not require RTSP server modifications; it only affects the client implementation.

Disadvantages:

*Requires a STUN server deployed in the public address space.

*Only works with NATs that perform endpoint independent and address dependent mappings. Port and address dependent filtering NATs create some issues.

*Brittle to NATs changing the properties of the NAT mapping and filtering.

*Does not work with port and address dependent mapping NATs without server modifications.

*Will mostly not work if a NAT uses multiple IP addresses, since RTSP server generally requires all media streams to use the same IP as used in the RTSP connection to prevent becoming a DDOS tool.

*Interaction problems exist when a RTSP-aware ALG interferes with the use of STUN for NAT traversal unless TLS secured RTSP message exchange is used.

*Using STUN requires that RTSP servers and clients support the updated RTSP specification, because it is no longer possible to guarantee that RTP and RTCP ports are adjacent to each other, as required by the "client_port" and "server_port" parameters in RFC2326.

Transition:
The usage of STUN can be phased out gradually as the first step of a STUN capable server or client should be to check the presence of NATs. The removal of STUN capability in the client implementations will have to wait until there is absolutely no need to use STUN.
For the "Embedded STUN" method the following applies:
Advantages:

*STUN is a solution first used by SIP applications. As shown above, with little or no changes, RTSP application can re-use STUN as a NAT traversal solution, avoiding the pit-fall of solving a problem twice.

*STUN has built-in message authentication features, which makes it more secure. See next section for an in-depth security discussion.

*This solution works as long as there is only one RTSP end point in the private address realm, regardless of the NAT's type. There may even be multiple NATs (see figure 1 in RFC3489).

*Compares to other UDP based NAT traversal methods in this document, STUN requires little new protocol development (since STUN is already a IETF standard), and most likely less implementation effort, since open source STUN server and client have become available [STUN-IMPL]. There is the need to embed STUN in RTSP server and client, which require a de-multiplexer between STUN packets and RTP/RTCP packets. There is also a need to register the proper feature tags.

Disadvantages:

*Some extensions to the RTSP core protocol, signaled by RTSP feature tags, must be introduced.

*Requires an embedded STUN server to co-locate on each of RTSP server's media protocol's ports (e.g. RTP and RTCP ports), which means more processing is required to de-multiplex STUN packets from media packets. For example, the de-multiplexer must be able to differentiate a RTCP RR packet from a STUN packet, and forward the former to the streaming server, the later to STUN server.

*Does not support use cases that requires the RTSP connection and the media reception to happen at different addresses, unless the servers sequrity policy is relaxed.

*Interaction problems exist when a RTSP ALG is not aware of STUN unless TLS is used to protect the RTSP messages.

*Using STUN requires that RTSP servers and clients support the updated RTSP specification, and they both agree to support the NAT traversal feature.

*Increases the setup delay with at least the amount of time it takes to perform STUN message exchanges. Most likely an extra SETUP sequence will be required.

Transition:
The usage of STUN can be phased out gradually as the first step of a STUN capable machine can be to check the presence of NATs for the presently used network connection. The removal of STUN capability in the client implementations will have to wait until there is absolutely no need to use STUN.

4.1.7. Security Considerations

To prevent RTSP server being used as Denial of Service (DoS) attack tools the RTSP Transport header parameter "destination" and "dest_addr" are generally not allowed to point to any IP address other than the one that RTSP message originates from. The RTSP server is only prepared to make an exception of this rule when the client is trusted (e.g., through the use of a secure authentication process, or through some secure method of challenging the destination to verify its willingness to accept the RTP traffic). Such restriction means that STUN does not work for use cases where RTSP and media transport goes to different address.
In terms of security property, STUN combined with destination address restricted RTSP has the same security properties as the core RTSP. It is protected from being used as a DoS attack tool unless the attacker has ability the to spoof the TCP connection carrying RTSP messages. Using STUN's support for message authentication and secure transport of RTSP messages, attackers cannot modify STUN responses or RTSP messages to change media destination. This protects against hijacking, however

as a client can be the initiator of an attack, these mechanisms cannot securely prevent RTSP servers being used as DoS attack tools.

4.2. ICE

4.2.1. Introduction

ICE (Interactive Connectivity Establishment) *[RFC5245]* is a methodology for NAT traversal that has been developed for SIP using SDP offer/answer. The basic idea is to try, in a parallel fashion, all possible connection addresses that an end point may have. This allows the end-point to use the best available UDP "connection" (meaning two UDP end-points capable of reaching each other). The methodology has very nice properties in that basically all NAT topologies are possible to traverse.
Here is how ICE works on a high level. End point A collects all possible address that can be used, including local IP addresses, STUN derived addresses, TURN addresses, etc. On each local port that any of these address and port pairs leads to, a STUN server is installed. This STUN server only accepts STUN requests using the correct authentication through the use of username and password.
End-point A then sends a request to establish connectivity with end-point B, which includes all possible destinations to get the media through too A. Note that each of A's published address/port pairs has a STUN server co-located. B, in its turn provides A with all its possible destinations for the different media streams. A and B then uses a STUN client to try to reach all the address and port pairs specified by A from its corresponding destination ports. The destinations for which the STUN requests have successfully completed are then indicated and selected.
If B fails to get any STUN response from A, all hope is not lost. Certain NAT topologies require multiple tries from both ends before successful connectivity is accomplished and therefore requests are retransmitted multiple times. The STUN requests may also result in that more connectivity alternatives are discovered and conveyed in the STUN responses.

4.2.2. Using ICE in RTSP

The usage of ICE for RTSP requires that both client and server be updated to include the ICE functionality. If both parties implement the necessary functionality the following steps could provide ICE support for RTSP.
This assumes that it is possible to establish a TCP connection for the RTSP messages between the client and the server. This is not trivial in scenarios where the server is located behind a NAT, and may require some TCP ports been opened, or the deployment of proxies, etc.
The negotiation of ICE in RTSP of necessity will work different than in SIP with SDP offer/answer. The protocol interactions are different and

thus the possibilities for transfer of states are also somewhat different. The goal is also to avoid introducing extra delay in the setup process at least for when the server is using a public address and the client is either having a public address or is behind NAT(s). This process is only intended to support PLAY mode, i.e. media traffic flows from server to client.

1. The ICE usage begins in the SDP. The SDP for the service indicates that ICE is supported at the server. No candidates can be given here as that would not work with the on demand, DNS load balancing, etc., that make a SDP indicate a resource on a server park rather than a specific machine.

2. The client gathers addresses and puts together its candidate for each media stream indicated in the session description.

3. In each SETUP request the client includes its candidates, promoting one for primary usage. This indicates for the server the ICE support by the client. One candidate is the primary candidate and here the prioritization for this address should be somewhat different compared to SIP. High performance rather than always successful is to recommended as it is most likely to be a server in the public.

4. The server responds to the SETUP (200 OK) for each media stream with its candidates. A server with a public address usually only provides a single ICE candidate. Also here one candidate is the server primary address.

5. The connectivity checks are performed. For the server the connectivity checks from the server to the clients have an additional usage. They verify that there is someone willingly to receive the media, thus protecting itself from performing unknowingly an DoS attack.

6. Connectivity checks from the client's primary to the server's primary was successful. Thus no further SETUP requests are necessary and processing can proceed with step 7. If another address than the primary has been verified by the client to work, that address may then be promoted for usage in a SETUP request (Goto 7). If the checks for the availble candidates failed and If further candidates have been derived during the connectivity checks, then those can be signalled in new candidate lines in SETUP request updating the list (Goto 5).

7. Client issues PLAY request. If the server also has completed its connectivity checks for this primary addresses (based on username as it may be derived addresses if the client was behind NAT) then it can directly answer 200 OK (Goto 8). If the

connectivity check has not yet completed it responds with a 1xx code to indicate that it is verifying the connectivity. If that fails within the set timeout an error is reported back. Client needs to go back to 6.

8. Process completed media can be delivered. ICE testing ports may be released.

To keep media paths alive the client needs to periodically send data to the server. This could be realized with either STUN or RTP No-op *[I-D.ietf-avt-rtp-no-op]* packets. RTCP sent by client should be able to keep RTCP open.

4.2.3. Implementation burden of ICE

The usage of ICE will require that a number of new protocols and new RTSP/SDP features be implemented. This makes ICE the solution that has the largest impact on client and server implementations amongst all the NAT/FW traversal methods in this document.
RTSP server implementation requirements are:

*STUN server features

*limited STUN client features

*SDP generation with more parameters.

*RTSP error code for ICE extension

RTSP client implantation requirements are:

*Limited STUN server features

*Limited STUN client features

*RTSP error code and ICE extension

4.2.4. Deployment Considerations

Advantages:

*Solves NAT connectivity discovery for basically all cases as long as a TCP connection between them can be established. This includes servers behind NATs. (Note that a proxy between address domains may be required to get TCP through).

*Improves defenses against DDOS attacks, as media receiving client requires authentications, via STUN on its media reception ports.

Disadvantages:

*Increases the setup delay with at least the amount of time it takes for the server to perform its STUN requests.

*Assumes that it is possible to de-multiplex between media packets and STUN packets.

*Has fairly high implementation burden put on both RTSP server and client.

4.2.5. Security Consideration

One should review the security consideration section of ICE and STUN to understand that ICE is contains some potential issues. However these can be avoided by a correctly utilizing ICE in RTSP. In fact ICE do help avoid the DDoS issue with RTSP substantially as it reduces the possibility for a DDoS using RTSP servers to attackers that are on-path between the RTSP server and the target and capable of intercepting the STUN connectivity check packets and correctly send a response to the server.

4.3. Symmetric RTP

4.3.1. Introduction

Symmetric RTP is a NAT traversal solution that is based on requiring RTSP clients to send UDP packets to the server's media output ports. Conventionally, RTSP servers send RTP packets in one direction: from server to client. Symmetric RTP is similar to connection-oriented traffic, where one side (e.g., the RTSP client) first "connects" by sending a RTP packet to the other side's RTP port, the recipient then replies to the originating IP and port.
Specifically, when the RTSP server receives the "connect" RTP packet (a.k.a. FW packet, since it is used to punch a hole in the FW/NAT and to aid the server for port binding and address mapping) from its client, it copies the source IP and Port number and uses them as delivery address for media packets. By having the server send media traffic back the same way as the client's packet are sent to the server, address mappings will be honored. Therefore this technique works for all types of NATs. However, it does require server modifications. Unless there is built-in protection mechanism, symmetric RTP is very vulnerable to DDOS attacks, because attackers can simply forge the source IP & Port of the binding packet. Using the rule for restriciting IP address to that one of the signalling connection will need to be applied here also.

4.3.2. Necessary RTSP extensions

To support symmetric RTP the RTSP signaling must be extended to allow the RTSP client to indicate that it will use symmetric RTP. The client

also needs to be able to signal its RTP SSRC to the server in its SETUP request. The RTP SSRC is used to establish some basic level of security against hijacking attacks. Care must be taken in choosing client's RTP SSRC. First, it must be unique within all the RTP sessions belonging to the same RTSP session. Secondly, if the RTSP server is sending out media packets to multiple clients from the same send port, the RTP SSRC needs to be unique amongst those clients' RTP sessions. Recognizing that there is a potential that RTP SSRC collision may occur, the RTSP server must be able to signal to client that a collision has occurred and that it wants the client to use a different RTP SSRC carried in the SETUP response or use unique ports per RTSP session. Using unique ports limits an RTSP server in the number of session it can simultaneously handle per interface IP addresses.

4.3.3. Deployment Considerations

Advantages:

*Works for all types of NATs, including those using multiple IP addresses. (Requirement 1 in Section 3).

*Have no interaction problems with any RTSP ALG changing the client's information in the transport header.

Disadvantages:

*Requires modifications to both RTSP server and client.

*Limited to work with servers that have an public IP address.

*The format of the RTP packet for "connection setup" (a.k.a FW packet) is yet to be defined. One possibility is to use RTP No-Op packet format in [I-D.ietf-avt-rtp-no-op].

*Has the same security situation as STUN and will need to use address restrictions.

4.3.4. Security Consideration

Symmetric RTP's major security issue is that RTP streams can be hijacked and directed towards any target that the attacker desires unless address restricitons are used.
The most serious security problem is the deliberate attack with the use of a RTSP client and symmetric RTP. The attacker uses RTSP to setup a media session. Then it uses symmetric RTP with a spoofed source address of the intended target of the attack. There is no defense against this attack other than restricting the possible bind address to be the same as the RTSP connection arrived on. This prevents symmetric RTP to be used in use cases that require differet addresses for media destination and signalling.

A hijack attack can also be performed in various ways. The basic attack is based on the ability to read the RTSP signaling packets in order to learn the address and port the server will send from and also the SSRC the client will use. Having this information the attacker can send its own NAT-traversal RTP packets containing the correct RTP SSRC to the correct address and port on the server. The destination of the packets is set as the source IP and port in these RTP packets.
Another variation of this attack is for a man in the middle to modify the RTP binding packet being sent by a client to the server by simply changing the source IP to the target one desires to attack.
One can fend off the first attack by applying encryption to the RTSP signaling transport. However, the second variation is impossible to defend against. As a NAT re-writes the source IP and port this cannot be authenticated, but authentication is required in order to protect against this type of DOS attack.
Yet another issues is that these attacks also can be used to deny the client the service he desire from the RTSP server completely. For a man in the middle capable of reading the signalling traffic or intercepting the binding packets can completely deny the client service by modifying or originating binding packets of itself.
The random nonce used in the binding packet determines how well symmetric RTP can fend off stream-hijacking performed by parties that are not "man-in-the-middle". This proposal uses the 32-bit RTP SSRC field to this effect. Therefore it is important that this field is derived with a non-predictable randomizer. It should not be possible by knowing the algorithm used and a couple of basic facts, to derive what random number a certain client will use.
An attacker not knowing the SSRC but aware of which port numbers that a server sends from can deploy a brute force attack on the server by testing a lot of different SSRCs until it finds a matching one. Therefore a server SHOULD implement functionality that blocks ports that receive multiple FW packets (i.e. the packet that is sent to the server for FW traversal) with different invalid SSRCs, especially when they are coming from the same IP/Port.
To improve the security against attackers the random tag's length could be increased. To achieve a longer random tag while still using RTP and RTCP, it will be necessary to develop RTP and RTCP payload formats for carrying the random tag.

4.3.5. A Variation to Symmetric RTP

Symmetric RTP requires a valid RTP format in the FW packet, which is the first packet that the client sends to the server to set up virtual RTP connection. There is currently no appropriate RTP packet format for this purpose, although the No-Op format is a proposal to fix the problem [I-D.ietf-avt-rtp-no-op]. There exists a RFC that discusses the implication of different type of packets as keep-alives for RTP [RFC6263] and its findings are very relevant to the FW packet.

Meanwhile, there has been FW traversal techniques deployed in the wireless streaming market place that use non-RTP messages as FW packets. This section attempts to summarize a subset of those solutions that happens to use a variation to the standard symmetric RTP solution. In this variation of symmetric RTP, the FW packet is a small UDP packet that does not contain RTP header. Hence the solution can no longer be called symmetric RTP, yet it employs the same technique for FW traversal. In response to client's FW packet, RTSP server sends back a similar FW packet as a confirmation so that the client can stop the so called "connection phase" of this NAT traversal technique. Afterwards, the client only has to periodically send FW packets as keep-alive messages for the NAT mappings.
The server listens on its RTP-media output port, and tries to decode any received UDP packet as FW packet. This is valid since an RTSP server is not expecting RTP traffic from the RTSP client. Then, it can correlate the FW packet with the RTSP client's session ID or the client's SSRC, and record the NAT bindings accordingly. The server then sends a FW packet as the response to the client.
The FW packet can contain the SSRC to identify the RTP stream, and can be made no bigger than 12 bytes, making it distinctively different from RTP packets, whose header size is 12 bytes.
RTSP signaling can be added to do the following:

1. Enables or disables such FW message exchanges. When the FW/NAT has an RTSP-aware ALG, it is possible to disable FW message exchange and let ALG works out the address and port mappings.

2. Configures the number of re-tries and the re-try interval of the FW message exchanges.

Such FW packets may also contain digital signatures to support three-way handshake based receiver authentications, so as to prevent DDoS attacks described before.
This approach has the following advantages when compared with the symmetric RTP approach:

1. There is no need to define RTP payload format for FW traversal, therefore it is simple to use, implement and administer (Requirement 4 in Section 3), although a binding protocol must be defined.

2. When properly defined, this kind of FW message exchange can also authenticate RTP receivers, so as to prevent DDoS attacks for dual-hosted RTSP client. By dual-hosted RTSP client we mean the kind that uses one "perceived" IP address for RTSP message exchange, and a different "perceived" IP address for RTP reception. (Requirement 5 in Section 3).

This approach has the following disadvantages when compared with the symmetric RTP approach:

1. RTP traffic is normally accompanied by RTCP traffic. This approach needs to rely on RTCP RRs and SRs to enable NAT traversal for RTCP endpoints, or use the same type of FW messages also for RTCP endpoints.

2. The server's sender SSRC for the RTP stream must be signaled in RTSP's SETUP response, in the Transport header of the RTSP SETUP response.

A solution with a 3-way handshaking and its own FW packets can be compared with ICE and have the following differencies:

*Only works for servers with public IP addresses compared to any type of server

*Is somewhat simpler to implement due to the avoidance of the ICE prioritization and checkboard mechanisms.

However, a 3-way binding protocol is very similar to using STUN in both directions as binding protocol. Using STUN would remove the need for implementing a new protocol.

4.4. Application Level Gateways

4.4.1. Introduction

An Application Level Gateway (ALG) reads the application level messages and performs necessary changes to allow the protocol to work through the middle box. However this behavior has some problems in regards to RTSP:

1. It does not work when the RTSP protocol is used with end-to-end security. As the ALG can't inspect and change the application level messages the protocol will fail due to the middle box.

2. ALGs need to be updated if extensions to the protocol are added. Due to deployment issues with changing ALGs this may also break the end-to-end functionality of RTSP.

Due to the above reasons it is NOT RECOMMENDED to use an RTSP ALG in NATs. This is especially important for NATs targeted to home users and small office environments, since it is very hard to upgrade NATs deployed in home or SOHO (small office/home office) environment.

4.4.2. Outline On how ALGs for RTSP work

In this section, we provide a step-by-step outline on how one should go about writing an ALG to enable RTSP to traverse a NAT.

1. Detect any SETUP request.

2. Try to detect the usage of any of the NAT traversal methods that replace the address and port of the Transport header parameters "destination" or "dest_addr". If any of these methods are used, the ALG SHOULD NOT change the address. Ways to detect that these methods are used are:

   *For embedded STUN, it would be watch for a feature tag, like "nat.stun". If any of those exists in the "supported", "proxy-require", or "require" headers of the RTSP exchange.

   *For non-embedded STUN and TURN based solutions: This can in some case be detected by inspecting the "destination" or "dest_addr" parameter. If it contains either one of the NAT's external IP addresses or a public IP address. However if multiple NATs are used this detection may fail. Remapping should only be done for addresses belonging to the NATs own private address space.

   Otherwise continue to the next step.

3. Create UDP mappings (client given IP/port <-> external IP/port) where needed for all possible transport specification in the transport header of the request found in (1). Enter the public address and port(s) of these mappings in transport header. Mappings SHALL be created with consecutive public port number starting on an even number for RTP for each media stream. Mappings SHOULD also be given a long timeout period, at least 5 minutes.

4. When the SETUP response is received from the server the ALG MAY remove the unused UDP mappings, i.e. the ones not present in the transport header. The session ID SHOULD also be bound to the UDP mappings part of that session.

5. If SETUP response settles on RTP over TCP or RTP over RTSP as lower transport, do nothing: let TCP tunneling to take care of NAT traversal. Otherwise go to next step.

6. The ALG SHOULD keep alive the UDP mappings belonging to the an RTSP session as long as: RTSP messages with the session's ID has been sent in the last timeout interval, or UDP messages are sent on any of the UDP mappings during the last timeout interval.

7. The ALG MAY remove a mapping as soon a TEARDOWN response has been received for that media stream.

4.4.3. Deployment Considerations

Advantage:

*No impact on either client or server

*Can work for any type of NATs

Disadvantage:

*When deployed they are hard to update to reflect protocol modifications and extensions. If not updated they will break the functionality.

*When end-to-end security is used the ALG functionality will fail.

*Can interfere with other type of traversal mechanisms, such as STUN.

Transition:
An RTSP ALG will not be phased out in any automatically way. It must be removed, probably through the removal of the NAT it is associated with.

4.4.4. Security Considerations

An ALG will not work when deployment of end-to-end RTSP signaling security. Therefore deployment of ALG will likely result in that clients located behind NATs will not use end-to-end security.

4.5. TCP Tunneling

4.5.1. Introduction

Using a TCP connection that is established from the client to the server ensures that the server can send data to the client. The connection opened from the private domain ensures that the server can send data back to the client. To send data originally intended to be transported over UDP requires the TCP connection to support some type of framing of the media data packets. Using TCP also results in that the client has to accept that real-time performance may no longer be possible. TCP's problem of ensuring timely deliver was the reasons why RTP was developed. Problems that arise with TCP are: head-of-line blocking, delay introduced by retransmissions, highly varying rate due to the congestion control algorithm.

4.5.2. Usage of TCP tunneling in RTSP

The RTSP core specification [I-D.ietf-mmusic-rfc2326bis] supports interleaving of media data on the TCP connection that carries RTSP signaling. See section 14 in [I-D.ietf-mmusic-rfc2326bis] for how to perform this type of TCP tunneling. There also exist another way of transporting RTP over TCP defined in Appendix C.2. For signaling and rules on how to establish the TCP connection in lieu of UDP, see appendix C.2 in [I-D.ietf-mmusic-rfc2326bis]. This is based on the framing of RTP over the TCP connection as described in RFC 4571 *[RFC4571]*.

4.5.3. Deployment Considerations

Advantage:

*Works through all types of NATs where server is in the open.

Disadvantage:

*Functionality needs to be implemented on both server and client.

*Will not always meet multimedia stream's real-time requirements.

Transition:
The tunneling over RTSP's TCP connection is not planned to be phased-out. It is intended to be a fallback mechanism and for usage when total media reliability is desired, even at the price of loss of real-time properties.

4.5.4. Security Considerations

The TCP tunneling of RTP has no known security problem besides those already presented in the RTSP specification. It is not possible to get any amplification effect that is desired for denial of service attacks due to TCP's flow control. A possible security consideration, when session media data is interleaved with RTSP, would be the performance bottleneck when RTSP encryption is applied, since all session media data also needs to be encrypted.

4.6. TURN (Traversal Using Relay NAT)

4.6.1. Introduction

Traversal Using Relay NAT (TURN) *[RFC5766]* is a protocol for setting up traffic relays that allows clients behind NATs and firewalls to receive incoming traffic for both UDP and TCP. These relays are controlled and have limited resources. They need to be allocated before usage. TURN allows a client to temporarily bind an address/port pair on the relay (TURN server) to its local source address/port pair, which is used to

contact the TURN server. The TURN server will then forward packets between the two sides of the relay. To prevent DOS attacks on either recipient, the packets forwarded are restricted to the specific source address. On the client side it is restricted to the source setting up the mapping. On the external side this is limited to the source address/port pair of the first packet arriving on the binding. After the first packet has arrived the mapping is "locked down" to that address. Packets from any other source on this address will be discarded. Using a TURN server makes it possible for a RTSP client to receive media streams from even an unmodified RTSP server. However the problem is those RTSP servers most likely restrict media destinations to no other IP address than the one RTSP message arrives. This means that TURN could only be used if the server knows and accepts that the IP belongs to a TURN server and the TURN server can't be targeted at an unknown address or also the RTSP connection is relayed through the same TURN server.

4.6.2. Usage of TURN with RTSP

To use a TURN server for NAT traversal, the following steps should be performed.

1. The RTSP client connects with RTSP server. The client retrieves the session description to determine the number of media streams. To avoid the issue with having RTSP connection and media traffic from different addresses also the TCP connection must be done through the same TURN server as the one in the next step. This will require the usage of TURN for TCP *[RFC6062]*.

2. The client establishes the necessary bindings on the TURN server. It must choose the local RTP and RTCP ports that it desires to receive media packets. TURN supports requesting bindings of even port numbers and continuous ranges.

3. The RTSP client uses the acquired address and port mappings in the RTSP SETUP request using the destination header. Note that the server is required to have a mechanism to verify that it is allowed to send media traffic to the given address. The server SHOULD include its RTP SSRC in the SETUP response.

4. Client requests that the Server starts playing. The server starts sending media packet to the given destination address and ports.

5. The first media packet to arrive at the TURN server on the external port causes "lock down"; then TURN server forwards the media packets to the RTSP client.

6. When media arrives at the client, the client should try to verify that the media packets are from the correct RTSP server, by matching the RTP SSRC of the packet. Source IP address of this packet will be that of the TURN server and can therefore not be used to verify that the correct source has caused lock down.

7. If the client notices that some other source has caused lock down on the TURN server, the client should create new bindings and change the session transport parameters to reflect the new bindings.

8. If the client pauses and media are not sent for about 75% of the mapping timeout the client should use TURN to refresh the bindings.

4.6.3. Deployment Considerations

Advantages:

*Does not require any server modifications.

*Works for any types of NAT as long as the server has public reachable IP address.

Disadvantage:

*Requires another network element, namely the TURN server.

*A TURN server for RTSP is may not scale since the number of sessions it must forward is proportional to the number of client media sessions.

*TURN server becomes a single point of failure.

*Since TURN forwards media packets, it necessarily introduces delay.

*An RTSP ALG MAY change the necessary destinations parameter. This will cause the media traffic to be sent to the wrong address.

Transition:
TURN is not intended to be phase-out completely, see chapter 11.2 of [RFC5766]. However the usage of TURN could be reduced when the demand for having NAT traversal is reduced.

4.6.4. Security Considerations

An eavesdropper of RTSP messages between the RTSP client and RTSP server will be able to do a simple denial of service attack on the

media streams by sending messages to the destination address and port present in the RTSP SETUP messages. If the attacker's message can reach the TURN server before the RTSP server's message, the lock down can be accomplished towards some other address. This will result in that the TURN server will drop all the media server's packets when they arrive. This can be accomplished with little risk for the attacker of being caught, as it can be performed with a spoofed source IP. The client may detect this attack when it receives the lock down packet sent by the attacker as being mal-formatted and not corresponding to the expected context. It will also notice the lack of incoming packets. See bullet 7 in Section 4.6.2.

The TURN server can also become part of a denial of service attack towards any victim. To perform this attack the attacker must be able to eavesdrop on the packets from the TURN server towards a target for the DOS attack. The attacker uses the TURN server to setup a RTSP session with media flows going through the TURN server. The attacker is in fact creating TURN mappings towards a target by spoofing the source address of TURN requests. As the attacker will need the address of these mappings he must be able to eavesdrop or intercept the TURN responses going from the TURN server to the target. Having these addresses, he can set up a RTSP session and starts delivery of the media. The attacker must be able to create these mappings. The attacker in this case may be traced by the TURN username in the mapping requests.

The first attack can be made very hard by applying transport security for the RTSP messages, which will hide the TURN servers address and port numbers from any eavesdropper.

The second attack requires that the attacker have access to a user account on the TURN server to be able set up the TURN mappings. To prevent this attack the server shall verify that the target destination accept this media stream.

5. Firewalls

Firewalls exist for the purpose of protecting a network from traffic not desired by the firewall owner. Therefore it is a policy decision if a firewall will let RTSP and its media streams through or not. RTSP is designed to be firewall friendly in that it should be easy to design firewall policies to permit passage of RTSP traffic and its media streams.

The firewall will need to allow the media streams associated with a RTSP session pass through it. Therefore the firewall will need an ALG that reads RTSP SETUP and TEARDOWN messages. By reading the SETUP message the firewall can determine what type of transport and from where the media streams will use. Commonly there will be the need to open UDP ports for RTP/RTCP. By looking at the source and destination addresses and ports the opening in the firewall can be minimized to the least necessary. The opening in the firewall can be closed after a TEARDOWN message for that session or the session itself times out.

Simpler firewalls do allow a client to receive media as long as it has sent packets to the target. Depending on the security level this can have the same behavior as a NAT. The only difference is that no address translation is done. To be able to use such a firewall a client would need to implement one of the above described NAT traversal methods that include sending packets to the server to open up the mappings.

6. Comparision of NAT traversal techniques

This section evaluates the techniques described above against the requirements listed in section Section 3.
In the following table, the columns correspond to the numbered requirements. For instance, the column under R1 corresponds to the first requirement in section Section 3: MUST work for all flavors of NATs. The rows represent the different FW traversal techniques. SymRTP is short for symmetric RTP, "V.SymRTP" is short for "variation of symmetric RTP" as described in section Section 4.3.5.
A Summary of the requirements are:

R1 Work for all flavors of NATs

R2 Most work with Firewalls, including them with ALGs

R3 Should have minimal impact on clients not behind NATs

R4 Should be simple to use, Implement and administrate.

R5 Should provide a mitigation against DDoS attacks

	R1	R2	R3	R4	R5
STUN	Yes	Yes	No	Maybe	No
ICE	Yes	Yes	No	No	Yes
SymRTP	Yes	Yes	Yes	Maybe	No
V. SymRTP	Yes	Yes	Yes	Yes	future
3-W SymRTP	Yes	Yes	Yes	Maybe	Yes
TURN	Yes	Yes	No	No	Yes

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

In preceding sessions we have discussed security merits of each and every NAT/FW traversal methods for RTSP discussed here. In summary, the presence of NAT(s) is a security risk, as a client cannot perform source authentication of its IP address. This prevents the deployment of any future RTSP extensions providing security against hijacking of sessions by a man-in-the-middle.
Each of the proposed solutions has security implications. Using STUN will provide the same level of security as RTSP with out transport level security and source authentications; as long as the server does not grant a client request to send media to different IP addresses. Using symmetric RTP will have a higher risk of session hijacking or denial of service than normal RTSP. The reason is that there exists a probability that an attacker is able to guess the random tag that the client uses to prove its identity when creating the address bindings. This can be solved in the variation of symmetric RTP (section 6.3.5) with authentication features. The usage of an RTSP ALG does not increase in itself the risk for session hijacking. However the deployment of ALGs as sole mechanism for RTSP NAT traversal will prevent deployment of encrypted end-to-end RTSP signaling. The usage of TCP tunneling has no known security problems. However it might provide a bottleneck when it comes to end-to-end RTSP signaling security if TCP tunneling is used on an interleaved RTSP signaling connection. The usage of TURN has severe risk of denial of service attacks against a client. The TURN server can also be used as a redirect point in a DDOS attack unless the server has strict enough rules for who may create bindings.

9. Acknowledgements

The author would also like to thank all persons on the MMUSIC working group's mailing list that has commented on this document. Persons having contributed in such way in no special order to this protocol are: Jonathan Rosenberg, Philippe Gentric, Tom Marshall, David Yon, Amir Wolf, Anders Klemets, and Colin Perkins. Thomas Zeng would also like to give special thanks to Greg Sherwood of PacketVideo for his input into this memo.
Section Section 1.1 contains text originally written for RFC 4787 by Francois Audet and Cullen Jennings.

10. References

, "

[RFC2119]	

	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC0768]	Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
[RFC0793]	Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
[RFC2326]	Schulzrinne, H., Rao, A. and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
[RFC3550]	Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
[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.
[RFC3022]	Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
[RFC3424]	Daigle, L., IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.
[RFC2588]	Finlayson, R., "IP Multicast and Firewalls", RFC 2588, May 1999.
[RFC2663]	Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.
[RFC4787]	Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
[RFC4566]	Handley, M., Jacobson, V. and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
[I-D.ietf-mmusic-rfc2326bis]	Schulzrinne, H, Rao, A, Lanphier, R, Westerlund, M and M Stiemerling, "Real Time Streaming Protocol 2.0 (RTSP)", Internet-Draft draft-ietf-mmusic-rfc2326bis-28, October 2011.
[RFC5766]	Mahy, R., Matthews, P. and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
[RFC4571]	Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, July 2006.

[I-D.ietf-avt-rtp-no-op]	Andreasen, F, "A No-Op Payload Format for RTP", Internet-Draft draft-ietf-avt-rtp-no-op-04, May 2007.
[RFC5389]	Rosenberg, J., Mahy, R., Matthews, P. and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
[RFC5245]	Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/ Answer Protocols", RFC 5245, April 2010.
[RFC6263]	Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.
[RFC6062]	Perreault, S. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, November 2010.
[STUN-IMPL]	Open Source STUN Server and Client, http:// www.vovida.org/applications/downloads/stun/ index.html", June 2007.

Authors' Addresses

Magnus Westerlund Westerlund Ericsson Farogatan 6 Stockholm, SE-164 80 Sweden Phone: +46 8 719 0000 EMail: magnus.westerlund@ericsson.com

Thomas Zeng Zeng EMail: thomas.zeng@gmail.com