Behave
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2008

J. Rosenberg
Cisco Systems
R. Mahy
Plantronics
C. Huitema
Microsoft

July 8, 2007

Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)
draft-ietf-behave-turn-04.txt

Status of this Memo

Copyright Notice

Abstract

This specification defines an extension of the Session Traversal Utilities for NAT (STUN) Protocol for asking the STUN server to relay packets towards a client. This extension, called Traversal Using Relays around NAT (TURN), is useful for elements behind NATs whose

mapping behavior is address and port dependent. The extension purposefully restricts the ways in which the relayed address can be used. In particular, it prevents users from running general purpose servers from ports obtained from the STUN server.

Table of Contents

1. Introduction

Session Traversal Utilities for NAT (STUN) [1] provides a suite of tools for facilitating the traversal of NAT. Specifically, it defines the Binding Request, which is used by a client to determine its reflexive transport address towards the STUN server. The reflexive transport address can be used by the client for receiving packets from peers, but only when the client is behind "good" NATs. In particular, if a client is behind a NAT whose mapping behavior [9] is address or address and port dependent (sometimes called "bad" NATs), the reflexive transport address will not be usable for communicating with a peer.

The only way to obtain a transport address that can be used for corresponding with a peer through such a NAT is to make use of a relay. The relay sits on the public side of the NAT, and allocates transport addresses to clients reaching it from behind the private side of the NAT. These allocated addresses are from interfaces on the relay. When the relay receives a packet on one of these allocated addresses, the relay forwards it toward the client.

This specification defines an extension of STUN, called TURN, that allows a client to request an address on the STUN server itself, so that the STUN server acts as a relay. To accomplish that, this extension defines a handful of new STUN requests and indications. The Allocate request is the most fundamental component of this set of extensions. It is used to provide the client with a transport address that is relayed through the STUN server. A transport address which relays through an intermediary is called a relayed transport

address.

Though a relayed address is highly likely to work when corresponding with a peer, it comes at high cost to the provider of the relay service. As a consequence, relayed transport addresses should only be used as a last resort. Protocols using relayed transport addresses should make use of mechanisms to dynamically determine whether such an address is actually needed. One such mechanism, defined for multimedia session establishment protocols, based on the offer/answer protocol in RFC 3264 [4], is Interactive Connectivity Establishment (ICE) [8].

The mechanism defined here was previously a standalone protocol called Traversal Using Relay NAT (TURN), and is now defined as an extension of STUN. A STUN server that supports these extensions can be called a 'STUN relay' or more simply a 'TURN server'.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [2].

3. Definitions

Relayed Transport Address: A transport address that terminates on a server, and is forwarded towards the client. The STUN Allocate Request can be used to obtain a relayed transport address, for example.

TURN client: A STUN client that implements this specification. It obtains a relayed transport address that it provides to a small number of peers (usually one).

TURN server: A STUN server that implements this specification. It relays data between a TURN client and its peer.

5-tuple: A combination of the source IP address and port,

destination IP address and port, and transport protocol (UDP, TCP, or TLS over TCP). It uniquely identifies a TCP connection, TLS channel, or bi-directional flow of UDP datagrams.

Permission: A record of an IP address and transport of a peer that is permitted to send traffic to the TURN client. The TURN server will only forward traffic to its client from peers that match an existing permission.

4. Overview of Operation

In a typical configuration, a TURN client is connected to a private network and through one or more NATs to the public Internet. On the public Internet is a TURN server. This specification defines several new messages and a new framing mechanism that add the ability for a STUN server to act as a packet relay. The text in this section explains the typical usage of this relay extension.

First the client sends an Allocate request to the server, which the server authenticates. The server generates an Allocate response with the allocated address, port, and target transport. All other STUN messages defined by this specification happen in the context of an allocation.

A successful Allocate Request just reserves an address on the TURN

server. Data does not flow through an allocated port until the TURN client asks the TURN server to open a permission. It can do this by sending data to the far end with a Send Indication for UDP allocations, by sending a ConnectRequest for TCP allocations, or by setting the default destination for either transport. While the client can request more than one permission per allocation, it needs to request each permission explicitly and one at a time. This insures that a client can't use a TURN server to run a traditional server, and partially protects the client from DoS attacks.

Once a permission is open, the client can then receive data flowing back from its peer. Initially this data is wrapped in a STUN Data Indication. Since multiple permissions can be open simultaneously, the Data Indication contains the Remote Address attribute so the TURN client knows which peer sent the data. The client can send data to

any of its peers with the Send Indication.

Once the client wants to primarily receive from one peer, it can send a SetActiveDestination request. All subsequent data received from the active peer is forwarded directly to the client and vice versa, except that it is wrapped or framed according to the protocol used between the TURN client and TURN server. The client can send subsequent SetActiveDestination requests to change or remove the active destination.

When the TURN client to server communication is over a datagram protocol (UDP), any datagram received from the active peer that has the STUN magic cookie is wrapped in a Data Indication. Likewise any datagram sent by the client that has the STUN magic cookie and is intended for the active peer is wrapped in a Send Indication. This wrapping prevents the STUN relay server from inappropriately interpreting end-to-end data.

Over stream-based transports (TCP and TLS over TCP), the TURN client and server always use some additional framing (defined in Section 5) so that end-to-end data is distinguishable from STUN control messages. This additional framing just has a type and a length field. The value of the type field was chosen so it is always distinguishable from an unframed STUN request or response.

The SetActiveDestination Request does not close other bindings. Data to and from other peers is still wrapped in Send and Data indications respectively.

Allocations can also request specific attributes such as the desired Lifetime of the allocation, and the maximum Bandwidth. Clients can also request specific port assignment behavior, for example, a specific port number, odd or even port numbers, or pairs of

sequential port numbers.

4.1. Transports

TURN clients can communicate with a TURN server using UDP, TCP, or TLS over TCP. A TURN can even relay traffic between two different transports with certain restrictions. A TURN can never relay from an unreliable transport (client to server) to a reliable transport to

the peer. Note that a TURN server never has a TLS relationship with a client's peer, since the TURN server does not interpret data above the TCP layer. When relaying data sent from a stream-based protocol to a UDP peer, the TURN server emits datagrams which are the same length as the length field in the STUN TCP framing or the length field in a Send Indication. Likewise, when a UDP datagram is relayed from a peer over a stream-based transport, the length of the datagram is the length of the TCP framing or Data Indication.

client to TURN	TURN to peer
UDP	UDP
TCP	TCP
TCP	UDP
TLS	TCP
TLS	UDP

For TURN clients, using TLS over TCP provides two benefits. When using TLS, the client can be assured that the address of the client's peers are not visible to an attacker except by traffic analysis downstream of the TURN server. Second, the client may be able to communicate with TURN servers using TLS that it would not be able to communicate with using TCP or UDP due to the configuration of a firewall between the TURN client and its server. TLS between the client and TURN server in this case just facilitates traversal.

For TCP connections, the Connection Request allows the client to ask the server to open a connection to the peer. This also adds a permission to accept an incoming TCP connection from the remote address of the peer. When the server and the peer try to open a TCP connection at the same time, this is called TCP simultaneous open.

When the TURN-to-peer leg is TCP, the TURN client needs to be aware of the status of these TCP connections. The TURN extension defines application states for a TCP connection as follows: LISTEN, ESTABLISHED, and CLOSED. Consequently, the TURN server sends a ConnectionState Indication for a binding whenever the relay connection status changes for one of the client's bindings, except

when the status changes due to a TURN client request (ex: an explicit

binding deallocation).

4.2. Tuple Terminology

To relay data to and from the correct location, the TURN server maintains an association between an internal address (called a 5-tuple) and one or more external 5-tuples, as shown in Figure 1. The internal 5-tuple identifies the path between the TURN client and the TURN server. It consists of the protocol (UDP, TCP, or TLS over TCP), the internal local IP address and port number and the source IP address and port number of the STUN client, as seen by the relay server. For example, for UDP, the internal 5-tuple is the combination of the IP address and port from which the STUN client sent its Allocate Request, with the IP address and port from which the corresponding Allocate Response was sent.

The external local transport address is the IP address and port allocated to the TURN client (the allocated transport address). The external 5-tuple is the combination of the external local transport address and the IP address and port of an external client that the STUN client is communicating with through the STUN server. Initially, there aren't any external 5-tuples, since the STUN client hasn't communicated with any other hosts yet. As packets are received on or sent from the allocated transport address, external 5-tuples are created.

> While the terminology used in this document refers to 5-tuples, the TURN server can store whatever identifier it likes that yields identical results. Specifically, many implementations may use a file-descriptor in place of a 5-tuple to represent a TCP connection.

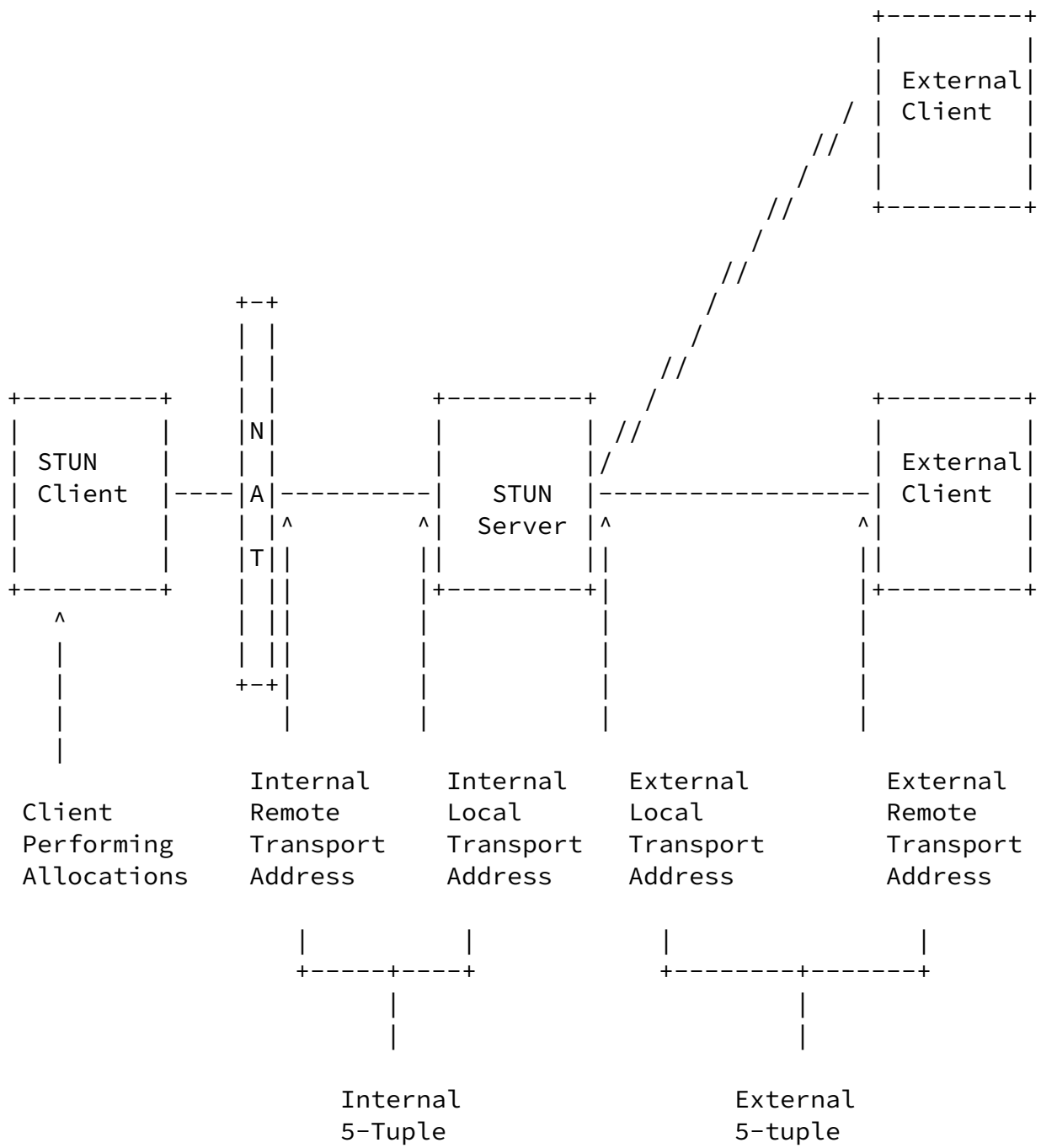


Figure 1

4.3. Keepalives

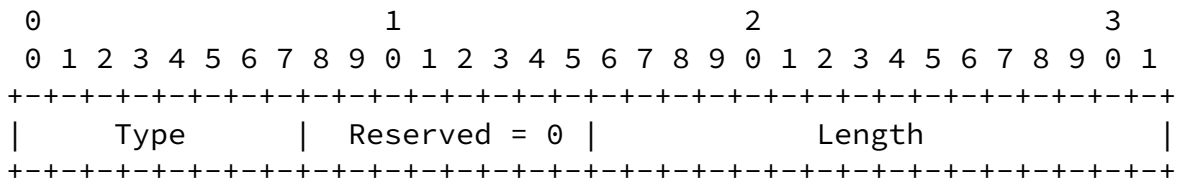
Since the main purpose of STUN and the relay extension are to traverse NATs, it is natural to consider which elements are responsible for generating sufficient periodic traffic to insure that NAT bindings stay alive. Relay clients need to send data frequently enough to keep both NAT bindings and the TURN server internal

permissions fresh. Like NAT bindings, the TURN server bindings are refreshed by ordinary data traffic relayed to and from the peer.

Unlike permissions, allocations on the TURN server have an explicit expiration time and need to be refreshed explicitly by the client. When an allocation expires, all permissions associated with that allocation are automatically deleted.

5. New Framing Mechanism for Stream-Oriented Transports

Over stream-based transports, the TURN client and server need to use additional framing so that end-to-end data is distinguishable from STUN control messages, and so that the TURN server can perform conversion from streams to datagrams and vice versa. This additional framing has a one octet type, one reserved octet, and a 2 octet length field. The first octet of this framing is 0x02 to indicate STUN messages or 0x03 to indicate end-to-end data to or from the active destination. Note that the first octet is always distinguishable from an unframed STUN request or response (which is always 0x00 or 0x01). The second octet is reserved and MUST be set to zero. The length field counts the number of octets immediately after the length field itself.



Use of this framing mechanism is discussed in Section 9 and Section 10.

6. New STUN Requests and Indications

This document defines three new requests (along with their success and error responses) and three indications. It also defines processing rules for the STUN server and client on receipt of non-STUN messages. See Section 9 and Section 10

The new messages are:

Request/Response Transactions
0x003 : Allocate
0x004 : Set Active Destination
0x005 : Connect

Indications
0x006 : Send
0x007 : Data
0x008 : Connect Status

In addition to STUN Messages (Requests, Responses, and Indications), TURN clients and servers send and receive non-STUN packets on the same ports used for STUN messages. How these entities distinguish STUN and non-STUN traffic is discussed in Section 9 and Section 10.

6.1. Allocate Request

6.1.1. Client Behavior

Client behavior for Allocate requests depends on whether the request is an initial one, for the purposes of obtaining a new relayed transport address, or a subsequent one, used for refreshing an existing allocation.

6.1.1.1. Initial Requests

When a client wishes to obtain a transport address, it sends an Allocate Request to the server. This request is constructed and sent using the general procedures defined in [1]. The server will challenge the request for credentials. The client MAY either provide its credentials to the server directly, or it MAY obtain a short-term set of credentials using the Shared Secret request and then use those

as the credentials in the Allocate request.

The client SHOULD include a BANDWIDTH attribute, which indicates the maximum bandwidth that will be used with this binding. If the maximum is unknown, the attribute is not included in the request.

The client MAY request a particular lifetime for the allocation by including it in the LIFETIME attribute in the request. The default lifetime is 10 minutes.

The client MAY include a REQUESTED-PORT-PROPS, REQUESTED-TRANSPORT, or REQUESTED-IP attribute in the request to obtain specific types of transport addresses. Whether these are needed depends on the application using the TURN server. As an example, the Real Time Transport Protocol (RTP) [3] requires that RTP and RTCP ports be an adajacent pair, even and odd respectively, for compatibility with a

previous version of that specification. The REQUESTED-PORT-PROPS attribute allows the client to ask the relay for those properties. The client MUST NOT request the TCP transport in an Allocate request sent to the TURN server over UDP.

Processing of the response follows the general procedures of [1]. A successful response will include both a RELAY-ADDRESS and an XOR-MAPPED-ADDRESS attribute, providing both a relayed transport address and a reflexive transport address, respectively, to the client. The server will expire the allocation after LIFETIME seconds have passed if not refreshed by another Allocate request. The server will allow the user to send and receive at least the amount of data indicated in the BANDWIDTH attribute per allocation. (At its discretion the server can optionally discard data above this threshold.)

If the response is an error response and contains a 442, 443 or 444 error code, the client knows that its requested properties could not be met. The client MAY retry with different properties, with the same properties (in a hope that something has changed on the server), or give up, depending on the needs of the application. However, if the client retries, it SHOULD wait 500ms, and if the request fails again, wait 1 second, then 2 seconds, and so on, exponentially backing off.

6.1.1.2. Subsequent Requests

Before 3/4 of the lifetime of the allocation has passed (the lifetime of the allocation is conveyed in the LIFETIME attribute of the Allocate Response), the client SHOULD refresh the allocation with another Allocate Request if it wishes to keep the allocation.

To perform a refresh, the client generates an Allocate Request as described in Section 6.1.1.1. If the initial request was authenticated with a shared secret P that the client holds with the server, or using a short term password derived from P through a Shared Secret request, the client MUST use shared secret P, or a short-term password derived from it, in the subsequent request.

In a successful response, the RELAY-ADDRESS contains the same transport address as previously obtained, indicating that the binding has been refreshed. The LIFETIME attribute indicates the amount of additional time the binding will live without being refreshed. Note that an error response does not imply that the binding has been expired, just that the refresh has failed.

If a client no longer needs an allocation, it SHOULD perform an explict deallocation. If the client wishes to explicitly remove the allocation because it no longer needs it, it generates a subsequent

Allocate request, but sets the LIFETIME attribute to zero. This will cause the server to remove the allocation, and all associated bindings. For connection-oriented transports such as TCP, the client can also remove the allocation (and all associated bindings) by closing the relevant connection with the TURN server.

6.1.2. Server Behavior

The server first processes the request according to the general request processing rules in [1]. This includes performing authentication, and checking for mandatory unknown attributes. Due to the fact that the STUN server is allocating resources for processing the request, Allocate requests MUST be authenticated, and furthermore, MUST be authenticated using either a shared secret known between the client and server, or a short term password derived from it.

Note that Allocate requests, like most other STUN requests, can be

sent to the TURN server over UDP, TCP, or TCP/TLS.

The behavior of the server when receiving an Allocate Request depends on whether the request is an initial one, or a subsequent one. An initial request is one whose source and destination transport address do not match the internal remote and local transport addresses of an existing internal 5-tuple. A subsequent request is one whose source and destination transport address matches the internal remote and local transport address of an existing internal 5-tuple.

6.1.2.1. Initial Requests

The server attempts to allocate transport addresses. It first looks for the BANDWIDTH attribute for the request. If present, the server determines whether or not it has sufficient capacity to handle a binding that will generate the requested bandwidth.

If it does, the server attempts to allocate a transport address for the client. The Allocate request can contain several additional attributes that allow the client to request specific characteristics of the transport address. First, the server checks for the REQUESTED-TRANSPORT attribute. This indicates the transport protocol requested by the client. This specification defines values for UDP and TCP.

> As a consequence of the REQUESTED-TRANSPORT attribute, it is possible for a client to connect to the server over TCP or TLS over TCP and request a UDP transport address. In this case, the server will relay data between the transports.

If the requested transport is supported, the server allocates a port using the requested transport protocol. If the REQUESTED-TRANSPORT attribute contains a value of the transport protocol unknown to the server, or known to the server but not supported by the server in the context of this request, the server MUST reject the request and include a 442 (Unsupported Transport Protocol) in the response, or redirect the request. If the request did not contain a REQUESTED-TRANSPORT attribute, the server MUST use the same transport protocol as the request arrived on.

Next, the server checks for the REQUESTED-IP attribute. If present,

it indicates a specific interface from which the client would like its transport address allocated. If this interface is not a valid one for allocations on the server, the server MUST reject the request and include a 443 (Invalid IP Address) error code in the response, or else redirect the request to a server that is known to support this IP address. If the IP address is one that is valid for allocations (presumably, the server is configured to know the set of IP addresses from which it performs allocations), the server MUST provide an allocation from that IP address. If the attribute is not present, the selection of an IP address is at the discretion of the server.

Finally, the server checks for the REQUESTED-PORT-PROPS attribute. If present, it indicates specific port properties desired by the client. This attribute is split into two portions: one portion for port behavior and the other for requested port alignment (whether the allocated port is odd, even, reserved as a pair, or at the discretion of the server).

If the port behavior requested is for a Specific Port, the server MUST attempt to allocate that specific port for the client. If the port is allocated to a different internal 5-tuple associated with the same STUN long-term credentials, the client is requesting a move. The server SHOULD replace the old internal 5-tuple with the new tuple over which this Allocate request arrived. The server MUST reject the move request if any of the attributes other than LIFETIME have changed (BANDWIDTH, REQUESTED-TRANSPORT, etc.).

If the specific port is not available (in use or reserved), the server MUST reject the request with a 444 (Invalid Port) response or redirect to an alternate server. For example, the STUN server could reject a request for a Specific Port because the port is temporarily reserved as part of an adjacent pair of ports, or because the requested port is a well-known port (1-1023).

If the client requests "even" port alignment, the server MUST attempt to allocate an even port for the client. If an even port cannot be obtained, the server MUST reject the request with a 444 (Invalid

Port) response or redirect to an alternate server. If the client requests odd port alignment, the server MUST attempt to allocate an odd port for the client. If an odd port cannot be obtained, the server MUST reject the request with a 444 (Invalid Port) response or

redirect to an alternate server. Finally, the "Even port with hold of the next higher port" alignment is similar to requesting an even port. It is a request for an even port, and MUST be rejected by the server if an even port cannot be provided, or redirected to an alternate server. However, it is also a hint from the client that the client will request the next higher port with a separate Allocate request. As such, it is a request for the server to allocate an even port whose next higher port is also available, and furthermore, a request for the server to not allocate that one higher port to any other request except for one that asks for that port explicitly. The server can honor this request for adjacency at its discretion. The only constraint is that the allocated port has to be even.

> Port alignment requests exist for compatibility with implementations of RTP which pre-date RFC 3550. These implementations use the port numbering conventions in (now obsolete) RFC 1889.

If any of the requested or desired constraints cannot be met, whether it be bandwidth, transport protocol, IP address or port, instead of rejecting the request, the server can alternately redirect the client to a different server that may be able to fulfill the request. This is accomplished using the 300 error response and ALTERNATE-SERVER attribute. If the server does not redirect and cannot service the request because the server has reached capacity, it sends a 507 (Insufficient Capacity) response. The server can also reject the request with a 486 (Allocation Quota Reached) if the user or client is not authorized to request additional allocations.

The server SHOULD only allocate ports in the range 1024-65535. This is one of several ways to prohibit relayed transport addresses from being used to attempt to run standard services. These guidelines are meant to be consistent with [9], since the relay is effectively a NAT.

Once the port is allocated, the server associates it with the internal 5-tuple and fills in that 5-tuple. The internal remote transport address of the internal 5-tuple is set to the source transport address of the Allocate Request. The internal local transport address of the internal 5-tuple is set to the destination transport address of the Allocate Request. For TCP, this amounts to associating the TCP connection from the TURN client with the allocated transport address.

If the Allocate request was authenticated using a shared secret between the client and server, this credential MUST be associated with the allocation. If the request was authenticated using a short term password derived from a shared secret, that shared secret MUST be associated with the allocation. This is used in all subsequent requests and indications to ensure that only the same client can use or modify the allocation it was given.

The allocation created by the Allocate request is also associated with a transport address, called the active destination. This transport address is used for forwarding data through the TURN server, and is described in more detail later. It is initially set to null when the allocation is created. In addition, the allocation created by the server is associated with a set of permissions. Each permission is a specific IP address identifying an external client. Initially, this list is null.

If the LIFETIME attribute was present in the request, and the value is larger than the maximum duration the server is willing to use for the lifetime of the allocation, the server MAY lower it to that maximum. However, the server MUST NOT increase the duration requested in the LIFETIME attribute. If there was no LIFETIME attribute, the server may choose a default duration at its discretion. In either case, the resulting duration is added to the current time, and a timer, called the allocation expiration timer, is set to fire at or after that time. Section 10.3 discusses behavior when the timer fires. Note that the LIFETIME attribute in the request can be zero. This typically happens for subsequent Allocations, and provides a mechanism to delete the allocation. It will force the immediate deletion of the allocation.

Once the port has been obtained and the activity timer started for the port binding, the server generates an Allocate Response using the general procedures defined in [1]. The transport address allocated to the client MUST be included in the RELAY-ADDRESS attribute in the response. In addition, this response MUST contain the XOR-MAPPED-ADDRESS attribute. This allows the client to determine its reflexive transport address in addition to a relayed transport address, from the same Allocate request.

The server MUST add a LIFETIME attribute to the Allocate Response. This attribute contains the duration, in seconds, of the allocation expiration timer associated with this allocation.

The server MUST add a BANDWIDTH attribute to the Allocate Response. This MUST be equal to the attribute from the request, if one was present. Otherwise, it indicates a per-binding cap that the server

is placing on the bandwidth usage on each binding. Such caps are

needed to prevent against denial-of-service attacks (See Section 12).

The server MUST add, as the final attribute of the request, a MESSAGE-INTEGRITY attribute. The key used in the HMAC MUST be the same as that used to validate the request.

6.1.2.2. Subsequent Requests

A subsequent Allocate request is one received whose source and destination IP address and ports match the internal 5-tuple of an existing allocation. The request is processed using the general server procedures in [1] and is processed identically to Section 6.1.2.1, with a few important exceptions.

First, the request MUST be authenticated using the same shared secret as the one associated with the allocation, or be authenticated using a short term password derived from that shared secret. If the request was authenticated but not with such a matching credential, the server MUST generate an Allocate Error Response with an appropriate error response code.

Secondly, if the allocated transport address given out previously to the client still matches the constraints in the request (in terms of request ports, IP addresses and transport protocols), the same allocation granted previously MUST be returned. However, if one of the constraints is not met any longer, because the client changed some aspect of the request, the server MUST free the previous allocation and allocate a new request to the client.

Finally, a subsequent Allocate request will set a new allocation expiration timer for the allocation, effectively canceling the previous lifetime expiration timer.

6.2. Procedures for all other Requests and Indications

Other than initial Allocate Requests, all requests and indications defined in this document need to be sent in the context of a valid allocation. The source and destination IP address and ports for these STUN messages MUST match the internal 5-tuple of an existing allocation. These processed using the general server procedures in

[1] with a few important additions. For requests, if there is no matching allocation, the server MUST generate a 437 (No Binding) Send Error Response. For indications, if there is no matching allocation, the indication is silently discarded.

All requests and indications MUST be authenticated using the same shared secret as the one associated with the allocation, or be authenticated using a short term password derived from that shared

secret. If the request was authenticated but not with such a matching credential, the server MUST generate an Allocate Error Response with an appropriate error response code, such as a 431 (Integrity Failure) or 436 (Unknown User).

6.3. Set Active Destination Request

6.3.1. Client Behavior

The Set Active Destination request allows the client to create an optimized relay function between the client and the server. When the server receives packets from a particular preferred external peer, the server will forward those packets towards the client without encapsulating them in a Data Indication. Similarly, the client can send non-STUN packets to the server without encapsulation in a Send Indication, and these packets are forwarded to the external peer. Sending and receiving data in unencapsulated form is critical for efficiency purposes. One of the primary use cases for the STUN relay extensions is in support of Voice over IP (VoIP), which uses very small UDP packets to begin with. The extra overhead of an additional layer of encapsulation is considered unacceptable.

The Set Active Destination request is used by the client to provide the identity of this preferred external peer. The Set Active Destination address MAY contain a REMOTE-ADDRESS attribute. This attribute, when present, provides the address of the preferred external peer to the server. When absent, it clears the value of the preferred external peer. As a convenience, if the client sets the REMOTE-ADDRESS attribute to a peer without a permission, the server will add the corresponding permission.

The client MUST NOT send a Set Active Destination request with a REMOTE-ADDRESS attribute over an unreliable link (ex: UDP) if an

active destination is already set for that allocation. If the client wishes to set a new active destination, it MUST wait until a successful response is received to a Set Destination Request removing the active destination. The client SHOULD then continue to wait for an additional period of up to 5 seconds until it is extremely unlikely that any data from the previous active destination might still arrive. Failure to wait could cause the client to receive and attribute late data forwarded by the TURN server to the wrong peer. The client MAY wait a shorter period of time if the application has built-in addressing (such as the RTP [3] Sender Source) that makes it unlikely the client would incorrectly attribute late data. [OPEN ISSUE: is this OK with the WG?]

Consider the case where the active destination is set, and the server is relaying packets towards the client. The client knows the IP address and port where the packets came from - the current value of the active destination. The client issues a Set Active Destination Request to change the active destination, and receives a response. A moment later, a data packet is received, not encapsulated in a STUN Data Indication. What is the source if this packet? Is it the active destination that existed prior to the Set Active Destination request, or the one after? If the transport between the client and the STUN server is not reliable, there is no way to know.

6.3.2. Server Behavior

The Set Active Destination Request is used by a client to set the forwarding destination of all data that is not encapsulated in STUN Send Indications. In addition, when a matching permission is present, all data received from that external peer will be forwarded to the STUN client without being encapsulated in a Data Indication.

If the Set Active Destination request does not contain a REMOTE-ADDRESS attribute, the value of the active destination is cleared. If the Set Active Destination request contains a REMOTE-ADDRESS attribute, and the active destination is not set, the active destination is set to that IP address and port. If an active destination is already set, and the request was received over a

reliable transport, the active destination is changed to the new value. If the active destination is already set and the request was received over UDP, the Set Active Destination request is rejected with a 439 Active Destination Already Set error response. This prevents the race condition described in the previous section.

If the server sets the active destination and there is no permission associated with the REMOTE-ADDRESS, the server adds the corresponding permission. Note that if the permission associated with the active destination becomes invalid, the server does not reset the active destination. The client is expected to do this explicitly.

6.4. Connect Request

The Connect Request is used by a client when it has obtained an allocated transport address that is TCP. The client can use the Connect Request to ask the server to open a TCP connection to a specified destination address included in the request.

6.4.1. Server Behavior

If the allocation is for a UDP port, the server MUST reject the request with a 445 (Operation for TCP Only) response. If the request does not contain a REMOTE-ADDRESS attribute, the server sends a 400 (Bad Request) Connect error response,.

If the request contains a REMOTE-ADDRESS attribute, the IP address contained within it is added to the permissions for this allocation, if it was not already present. This happens regardless of whether the subsequent TCP connection attempt succeeds or not.

If a connection already exists for this address and port, the server returns a 446 (Connection Already Exists) Connect error response. Otherwise the server tries to establish the corresponding TCP connection and returns a Connect Success Response. This just indicates that the server added the permission and is attempting to establish a TCP connection. The server does not wait for the connection attempt to succeed or fail. The status of the connection

attempt is returned via the Connect Status Indication.

Note that the server needs to use the same source connection address for all connections/permissions associated with an allocation. For servers written using Berkeley sockets, the SO_REUSEADDR flag is typically used to use the same local address with multiple sockets.

6.5. Connection Status Indication

When the TURN to peer leg is TCP, the TURN client needs to be aware of the status of these TCP connections. The TURN extension defines application states for a TCP connection as follows: LISTEN, ESTABLISHED, CLOSED. Consequently, the TURN server sends a Connection State Indication for a TCP permission whenever the relay connection status changes for one of the client's permissions except when the status changes due to a TURN client request (ex: an explicit binding close or deallocation).

A TURN can only relay to a peer over TCP if the client communicates with the server over TCP or TLS over TCP. Because of this, the server can be assured that Connection Status Indications are received reliably.

6.6. Send Indication

6.6.1. Client Behavior

The Send Indication is used to ask the relay to forward data to a peer. It is typically used to send to a peer other than the active destination. For TCP allocated transport addresses, the client needs to wait for the peer to open a connection to the TURN server before it can send data. Data sent with a Send request prior to the opening of a TCP connection is discarded silently by the server.

The Send Indication MUST contain a REMOTE-ADDRESS attribute, which contains the IP address and port that the data is being sent to. The DATA attribute MAY be present, and contains the data that is to be

sent towards REMOTE-ADDRESS. If absent, the server will send an empty UDP packet in the case of UDP. In the case of TCP, the server will do nothing.

Since Send is an Indication, it generates no response. The client must rely on application layer mechanisms to determine if the data was received by the peer.

> Note that Send Indications are not authenticated and do not contain a MESSAGE-INTEGRITY attribute. Just like non-relayed data sent over UDP or TCP, the authenticity and integrity of this data can only be assured using security mechanisms at higher layers.

6.6.2. Server Behavior

A Send Indication is sent by a client after it has completed its Allocate transaction, in order to create permissions in the server and send data to an external client.

If a Send Indication contains no REMOTE-ADDRESS, the indication is discarded. If there is no DATA attribute, and the corresponding allocation is for TCP, the indication is discarded.

If the allocation is a UDP allocation, the server creates a UDP packet whose payload equals that content. The server sets the source IP address of the packet equal to the allocated transport address. The destination transport address is set to the contents of the REMOTE-ADDRESS attribute. If a permission does not exist for this destination the server creates one for this allocation. The server then sends the UDP packet. Note that any retransmissions of this packet which might be needed are not handled by the server. It is the responsibility of the client to generate another Send indication if needed.

If the allocation is a TCP allocation, the server checks if it has an existing TCP connection open from the allocated transport address to

the address in the REMOTE-ADDRESS attribute. If so, the server extracts the content of the DATA attribute and sends it over the matching TCP connection. If the server doesn't have an existing TCP connection to the destination, it adds the REMOTE-ADDRESS to the permission list and discards the data.

6.7. Data Indication

6.7.1. Client Behavior

Once a client has obtained an allocation and created permissions for a particular external client, the server can begin to relay packets from that external client towards the client. If the external client is not the active destination, this data is relayed towards the client in encapsulated form using the Data Indication.

The Data Indication contains two attributes - DATA and REMOTE-ADDRESS. The REMOTE-ADDRESS attribute indicates the source transport address that the request came from, and it will equal the external remote transport address of the external peer. When processing this data, a client MUST treat the data as if it came from this address, rather than the stun server itself. The DATA attribute contains the data from the UDP packet or TCP segment that was received. Note that the TURN server will not retransmit this indication over UDP.

> Note that Data Indications are not authenticated and do not contain a MESSAGE-INTEGRITY attribute. Just like non-relayed data sent over UDP or TCP, the authenticity and integrity of this data can only be assured using security mechanisms at higher layers.

6.7.2. Server Behavior

A server MUST send data packets towards the client using a Data Indication under the conditions described in Section 10.1. Data Indications MUST contain a DATA attribute containing the data to send, and MUST contain a REMOTE-ADDRESS attribute indicating where the data came from.

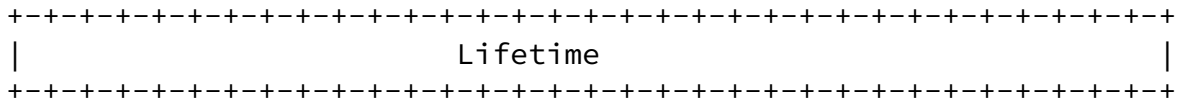
7. New Attributes

This STUN extension defines the following new attributes:

0x000D: LIFETIME
0x0010: BANDWIDTH
0x0012: REMOTE-ADDRESS
0x0013: DATA
0x0016: RELAY-ADDRESS
0x0018: REQUESTED-PORT-PROPS
0x0019: REQUESTED-TRANSPORT
0x0022: REQUESTED-IP
0x0021: TIMER-VAL
0x0023: CONNECT_STAT

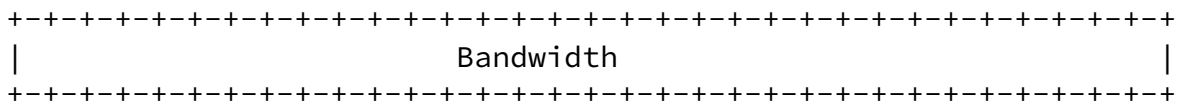
7.1. LIFETIME

The lifetime attribute represents the duration for which the server will maintain an allocation in the absence of data traffic either from or to the client. It is a 32 bit value representing the number of seconds remaining until expiration.



7.2. BANDWIDTH

The bandwidth attribute represents the peak bandwidth, measured in kbits per second, that the client expects to use on the binding in each direction.



7.3. REMOTE-ADDRESS

The REMOTE-ADDRESS specifies the address and port of the peer as seen from the TURN server. It is encoded in the same way as MAPPED-ADDRESS.

7.4. DATA

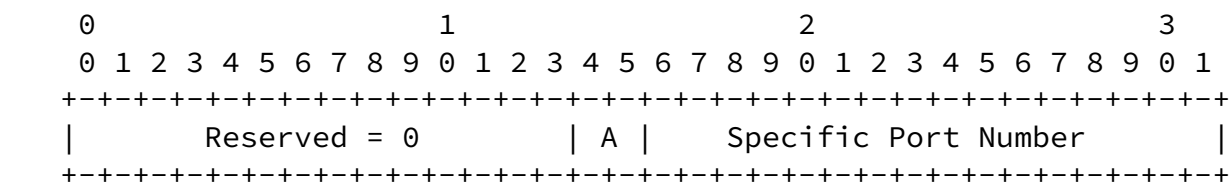
The DATA attribute is present in Send Indications and Data Indications. It contains raw payload data that is to be sent (in the case of a Send Request) or was received (in the case of a Data Indication). It is padded with zeros if its length is not divisible evenly by 4 octets

7.5. RELAY-ADDRESS

The RELAY-ADDRESS is present in Allocate responses. It specifies the address and port that the server allocated to the client. It is encoded in the same way as MAPPED-ADDRESS.

7.6. REQUESTED-PORT-PROPS

This attribute allows the client to request certain properties for the port that is allocated by the server. The attribute can be used with any transport protocol that has the notion of a 16 bit port space (including TCP and UDP). The attribute is 32 bits long. Its format is:



The two bits labeled A in the diagram above are for requested port alignment and have the following meaning:

00 no specific port alignment
01 odd port number
10 even port number
11 even port number; reserve next higher port

If the Specific Port Number field is zero, this means that no specific port is requested. If a specific port number is requested the value will be in the two low order octets. All other bits in this attribute are reserved and MUST be set to zero.

Even Port is a request to the server to allocate a port with even parity. The port filter is not used with this property. Odd Port is a request to the server to allocate a port with odd parity. The port filter is not used with this property. Even port, with a hold on the next higher port, is a request to the server to allocate an even port. Furthermore, the client indicates that it will want the next higher port as well. As such, the client requests that the server, if it can, not allocate the next higher port to anyone unless that

port is explicitly requested, which the client will itself do. The port filter is not used with this property. Finally, the Specific Port property is a request for a specific port. The port that is requested is contained in the Port filter.

7.7. REQUESTED-TRANSPORT

This attribute is used by the client to request a specific transport protocol for the allocated transport address. It is a 32 bit unsigned integer. Its values are:

0x0000 0000: UDP
0x0000 0001: TCP

If an Allocate request is sent over TCP and requests a UDP allocation, or an Allocate request is sent over TLS over TCP and requests a UDP or TCP allocation, the server will relay data between the two transports.

Extensions to TURN can define additional transport protocols in an IETF-consensus RFC.

7.8. REQUESTED-IP

The REQUESTED-IP attribute is used by the client to request that a specific IP address be allocated to it. This attribute is needed since it is anticipated that TURN servers will be multi-homed so as to be able to allocate more than 64k transport addresses. As a consequence, a client needing a second transport address on the same interface as a previous one can make that request.

The format of this attribute is identical to MAPPED-ADDRESS. However, the port component of the attribute is ignored by the server. If a client wishes to request a specific IP address and port, it uses both the REQUESTED-IP and REQUESTED-PORT-PROPS attributes.

7.9. CONNECT_STAT

This attribute is used by the server to convey the status of server-to-peer connections. It is a 32 bit unsigned integer. Its values are:

0x0000 0000: LISTEN
0x0000 0001: ESTABLISHED
0x0000 0002: CLOSED

8. New Error Response Codes

This document defines the following new Error response codes:

437 (No Binding): A request was received by the server that requires an allocation to be in place. However, there is none yet in place.

439 (Active Destination Already Set): A Set Active Destination request was received by the server over UDP. However, the active destination is already set to another value. The client should reset the active destination, wait for the hold-down period, and set the active destination to the new value.

442 (Unsupported Transport Protocol): The Allocate request asked for a transport protocol to be allocated that is not supported by the server.

443 (Invalid IP Address): The Allocate request asked for a transport address to be allocated from a specific IP address that is not valid on the server.

444 (Invalid Port): The Allocate request asked for a port to be allocated that is not available on the server.

445 (Operation for TCP Only): The client tried to send a request to perform a TCP-only operation on an allocation, and allocation is UDP.

446 (Connection Already Exists): The client tried to open a connection to a peer, but a connection to that peer already exists.

486 (Allocation Quota Reached): The user or client is not authorized to request additional allocations.

507 (Insufficient Capacity): The server cannot allocate a new port for this client as it has exhausted its relay capacity.

9. Client Procedures

9.1. Receiving and Sending Unencapsulated Data

Once the active destination has been set, a client will receive both STUN and non-STUN data on the socket on which the Allocate Request was sent. The encapsulation behavior depends on the transport protocol used between the STUN client and the TURN server.

9.1.1. Datagram Protocols

If the allocation was over UDP, datagrams which contain the STUN magic cookie are treated as STUN requests. All other data is non-STUN data, which MUST be processed as if it had a source IP address and port equal to the value of the active destination.

If the client wants to send data to the peer which contains the magic cookie in the same location as a STUN request, it MUST send that data encapsulated in a Send Indication, even if the active destination is set.

In addition, once the active destination has been set, the client can send data to the active destination by sending the data unencapsulated on that same socket. Unencapsulated data MUST NOT be sent if no active destination is set. Of course, even if the active destination is set, the client can send data to that destination at any time by using the Send Indication.

9.1.2. Stream Transport Protocols

If the allocation was over TCP or TLS over TCP, the client will receive data framed as described in Section 5. The client MUST treat data encapsulated as data with this framing as if it originated from the active destination.

For the any of the methods defined in this document, the client always sends data encapsulated using this framing scheme. It SHOULD frame data to the active destination as data or it MAY place the data inside a Send Indications and frame this as STUN traffic.

10. Server Procedures

Besides the processing of the request and indications described above, this specification defines rules for processing of data packets received by the STUN server. There are two cases - receipt of any packets on an allocated address, and receipt of non-STUN data on its internal local transport address.

10.1. Receiving Data on Allocated Transport Addresses

10.1.1. TCP Processing

If a server receives a TCP connection request on an allocated TCP transport address, it checks the permissions associated with that allocation. If the source IP address of the TCP SYN packet matches one of the permissions (the source port does not need to match), the

TCP connection is accepted. Otherwise, it is rejected. When a TCP connection is accepted, the server sends the corresponding client a Connect Status Indication with the CONNECT_STAT attribute set to ESTABLISHED. No information is passed to the client if the server rejects the connection because there is no corresponding permission.

If a server receives data on a TCP connection that terminates on the allocated TCP transport address, the server checks the value of the active destination. If it equals the source IP address and port of the data packet (in other words, if the active destination identifies the other side of the TCP connection), the data is taken from the TCP connection and sent towards the client in unencapsulated form. Otherwise, the data is sent towards the client in a Data Indication, also known as encapsulated form. In this form, the server MUST add a

REMOTE-ADDRESS which corresponds to the external remote transport address of the TCP connection, and MUST add a DATA attribute containing the data received on the TCP connection.

Note that, because data is forwarded blindly across TCP bindings, TLS will successfully operate over a TURN allocated TCP port if the linkage to the client is also TCP.

10.1.2. UDP Processing

If a server receives a UDP packet on an allocated UDP transport address, it checks the permissions associated with that allocation. If the source IP address of the UDP packet matches one of the permissions (the source port does not need to match), the UDP packet is accepted. Otherwise, it is discarded. If the packet is accepted, it is forwarded to the client. It will be forwarded in either encapsulated or unencapsulated form.

If the client to server communication is via UDP, the server looks for the existence of the STUN magic cookie in the data received from the peer. If the data contains the magic cookie, the server encapsulates the data in a Data Indication, sets the REMOTE_ADDRESS attribute, and forwards the indication to the client. If the magic cookie is not present, the server checks if the peer is the active destination. If so the data is forwarded unencapsulated, directly to the client. Otherwise the server encapsulates the data in a Data Indication, sets the REMOTE_ADDRESS and forwards to the client.

If the client to server communication is via TCP or TLS, the server checks if the peer is the active destination. If so, the data from the peer is framed as Data and sent to the client over the client to server connection. Otherwise, the server encapsulates the data in a Data Indication, sets the REMOTE_ADDRESS attribute, frames the indication as STUN traffic, and sends the indication over the

connection to the client. If the TCP connection generates an error (because, for example, the incoming UDP packet rate exceeds the throughput of the TCP connection), the data is discarded silently by the server.

10.2. Receiving Data on Internal Local Transport Addresses

If a server receives non-STUN UDP data from the client on its internal local transport address, and it is coming from an internal remote transport address associated with an existing allocation, it represents UDP data that the client wishes to forward. If there is no allocation associated with the source IP address and port number, or if there is an associated allocation but the active destination is not set, the server MUST discard the packet. If the active destination is set, the server places the data from the client in a UDP payload, and sets the destination address and port to the active destination. The UDP packet is then sent with a source IP address and port equal to the allocated transport address. Note that the server will not retransmit the UDP datagram.

If a server receives framed data on a TCP connection from a client, the server retrieves the allocation bound to that connection. If the active destination for the allocation is not set, the server MUST discard the data and close the connection. If the active destination is set, and the allocated transport protocol is TCP, the server forwards the data over the connection to the active destination. The data is then sent over that connection. If the connection is not established or if the transmission fails due to a TCP error, the data is discarded silently by the server. If the active destination is set, and the allocated transport protocol is UDP, the server places the data from the client in a UDP payload, and sets the destination address and port to the active destination. The UDP packet is then sent with a source IP address and port equal to the allocated transport address. Note that the server will not retransmit the UDP datagram.

If a TCP connection from a client is closed, the associated allocation is destroyed. This involves terminating any TCP connections from the allocated transport address to external peer (applicable only when the allocated transport address was TCP), and then freeing the allocated transport address (and all associated state maintained by the server) for use by other clients.

10.3. Lifetime Expiration

When the allocation expiration timer for a binding fires, the server MUST destroy the allocation. This involves terminating any TCP connections from the allocated transport address to external peers

(applicable only when the allocated transport address was TCP), and then freeing the allocated transport address (and all associated state maintained by the server) for use by other clients. A suggested value for the allocation expiration timer is 10 minutes.

The server is also expected to run a permission inactivity timer for each permission associated with an Allocation. If no traffic from the client is received, the permission inactivity timer will eventually expire and the server MUST delete the permission. A suggested value for the permission inactivity timer for UDP allocations is 60 seconds.

11. Client Discovery of TURN Servers

The STUN relay extensions differ from the binding requests defined in [1] in that they demands substantial resources from the STUN server. In addition, it seems likely that administrators might want to block connections from clients to the STUN server for relaying separately from connections for the purposes of binding discovery. As a consequence, TURN is expected to typically run on a separate port from basic STUN. The client discovers the address and port of the TURN server using the same DNS procedures defined in [1], but using an SRV service name of "stun-relay" instead of just "stun".

For example, to find TURN servers in the example.com domain, the TURN client performs a lookup for '_stun-relay._udp.example.com', '_stun-relay._tcp.example.com', and '_stun-relay._tls.example.com' if the STUN client wants to communicate with the TURN server using UDP, TCP, or TLS over TCP, respectively. The client assumes that all permissable transport protocols are supported from the TURN server to the peer for the client to server protocol selected.

> The STUN server is designed so the relay usage can run on a separate source port from non-relay usages. Since the client looks up the port number for the relay usage separately, servers can be configured to rely on this property. The STUN server MAY accept both relay and non-relay usages on the same port number, in which case it uses framing hints and choice of STUN messages to detect the STUN usage in use by a specific client.

12. Security Considerations

STUN servers implementing the TURN extensions allocate bandwidth and port resources to clients, in contrast to the Binding method defined in [1]. Therefore, a STUN server providing the relay usage requires authentication and authorization of STUN requests. This

authentication is provided by mechanisms defined in the STUN specification itself. In particular, digest authentication and the usage of short-term passwords, obtained through a digest exchange over TLS, are available. The usage of short-term passwords ensures that the Allocate Requests, which often do not run over TLS, are not susceptible to offline dictionary attacks that can be used to guess the long lived shared secret between the client and the server.

Because TURN servers allocate resources, they can be susceptible to denial-of-service attacks. All Allocate Requests are authenticated, so that an unknown attacker cannot launch an attack. An authenticated attacker can generate multiple Allocate Requests, however. To prevent a single malicious user from allocating all of the resources on the server, it is RECOMMENDED that a server implement a modest per user cap on the amount of bandwidth that can be allocated. Such a mechanism does not prevent a large number of malicious users from each requesting a small number of allocations. Attacks as these are possible using botnets, and are difficult to detect and prevent. Implementors of TURN should keep up with best practices around detection of anomalous botnet attacks.

A client will use the transport address learned from the RELAY-ADDRESS attribute of the Allocate Response to tell other users how to reach them. Therefore, a client needs to be certain that this address is valid, and will actually route to them. Such validation occurs through the message integrity checks provided in the Allocate response. They can guarantee the authenticity and integrity of the allocated addresses. Note that TURN is not susceptible to the attacks described in Section 12.2.3, 12.2.4, 12.2.5 or 12.2.6 of RFC 3489 [[TODO: Update references once 3489bis is more stable]]. These attacks are based on the fact that a STUN server mirrors the source IP address, which cannot be authenticated. STUN does not use the source address of the Allocate Request in providing the RELAY-ADDRESS, and therefore, those attacks do not apply.

TURN cannot be used by clients for subverting firewall policies. TURN has fairly limited applicability, requiring a user to send a packet to a peer before being able to receive a packet from that peer. This applies to both TCP and UDP. Thus, it does not provide a general technique for externalizing TCP and UDP sockets. Rather, it has similar security properties to the placement of an address-restricted NAT in the network, allowing messaging in from a peer only if the internal client has sent a packet out towards the IP address

of that peer. This limitation means that TURN cannot be used to run web servers, email servers, SIP servers, or other network servers that service a large number of clients. Rather, it facilitates rendezvous of NATted clients that use some other protocol, such as SIP, to communicate IP addresses and ports for communications.

Confidentiality of the transport addresses learned through Allocate requests does not appear to be that important, and therefore, this capability is not provided.

Relay servers are useful even for users not behind a NAT. They can provide a way for truly anonymous communications. A user can cause a call to have its media routed through a STUN server, so that the user's IP addresses are never revealed.

TCP transport addresses allocated by Allocate requests will properly work with TLS and SSL. However, any relay addresses learned through an Allcoate will not operate properly with IPSec Authentication Header (AH) [5] in transport mode. IPSec ESP [6] and any tunnel-mode ESP or AH should still operate.

13. IANA Considerations

This specification defines several new STUN messages, STUN attributes, and STUN response codes. This section directs IANA to add these new protocol elements to the IANA registry of STUN protocol elements.

13.1. New STUN Requests, Responses, and Indications

```
Request/Response Transactions
0x003  :  Allocate
0x004  :  Set Active Destination
0x005  :  Connect

Indications
0x006  :  Send
0x007  :  Data
0x008  :  Connect Status
```

13.2. New STUN Attributes

```
0x000D: LIFETIME
0x0010: BANDWIDTH
0x0012: REMOTE-ADDRESS
0x0013: DATA
0x0016: RELAY-ADDRESS
0x0018: REQUESTED-PORT-PROPS
0x0019: REQUESTED-TRANSPORT
0x0022: REQUESTED-IP
0x0021: TIMER-VAL
0x0023: CONNECT_STAT
```

13.3. New STUN response codes

```
437    No Binding
439    Active Destination Already Set
442    Unsupported Transport Protocol
443    Invalid IP Address
444    Invalid Port
445    Operation for TCP Only
446    Connection Already Exists
486    Allocation Quota Reached
507    Insufficient Capacity
```

14. IAB Considerations

The IAB has studied the problem of "Unilateral Self Address Fixing", which is the general process by which a client attempts to determine its address in another realm on the other side of a NAT through a collaborative protocol reflection mechanism RFC 3424 [7]. The TURN extension is an example of a protocol that performs this type of function. The IAB has mandated that any protocols developed for this purpose document a specific set of considerations.

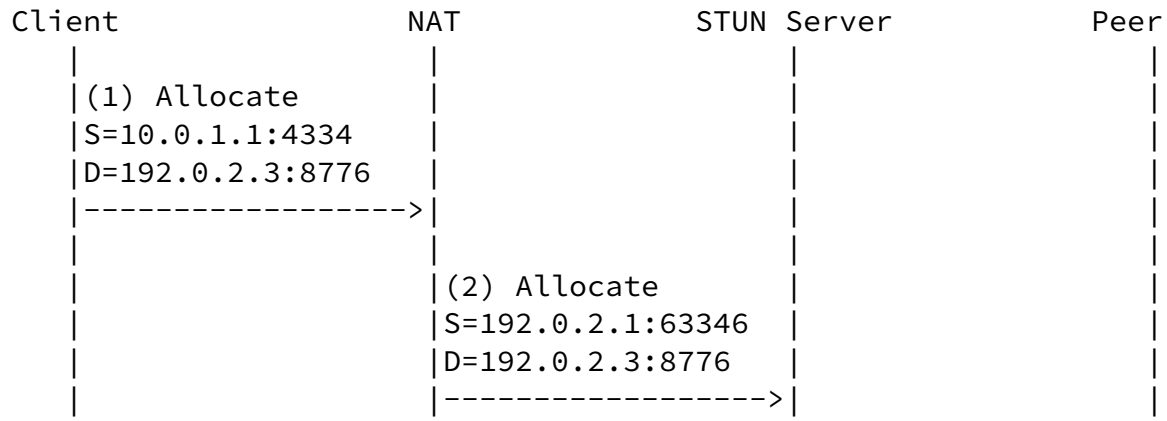
TURN is an extension of the STUN protocol. As such, the specific usages of STUN that use the TURN extensions need to specifically address these considerations. Currently the only STUN usage that uses TURN is ICE [8].

15. Example

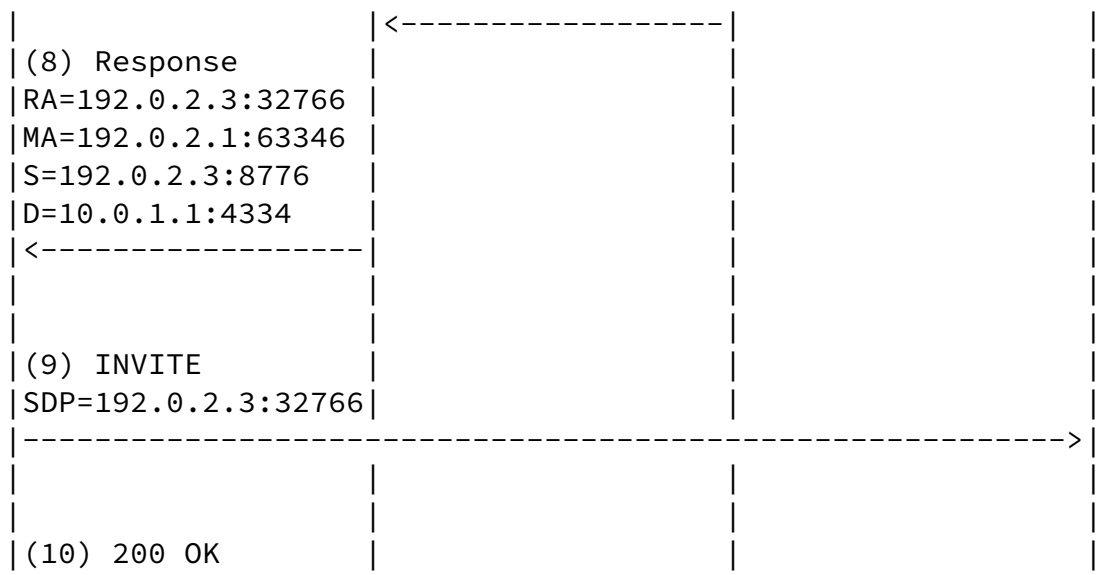
In this example, a client is behind a NAT. The client has a private address of 10.0.1.1. The STUN server is on the public side of the NAT, and is listening for TURN requests on 192.0.2.3:8776. The

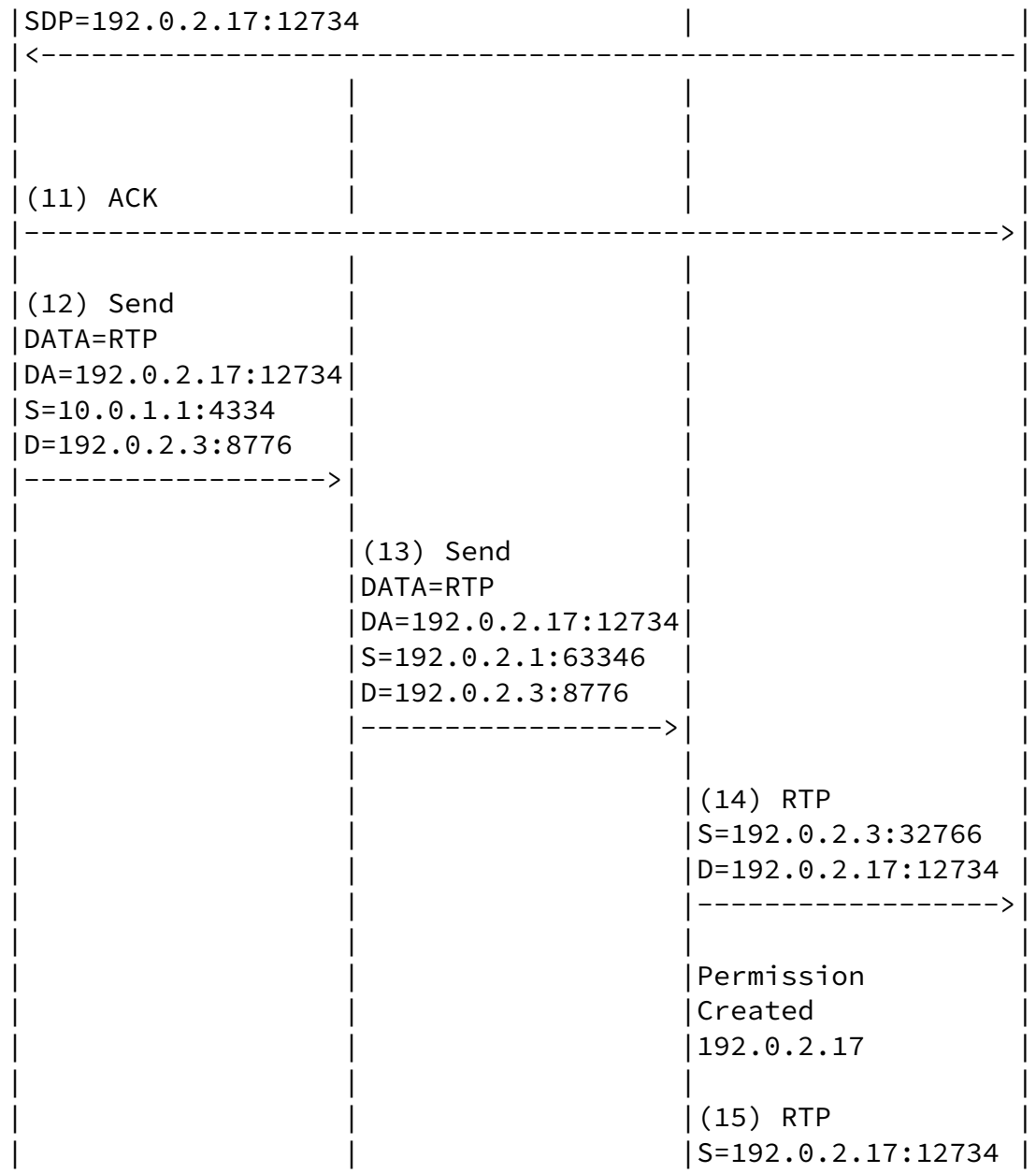
public side of the NAT has an IP address of 192.0.2.1. The client is attempting to send a SIP INVITE to a peer, and wishes to allocate an IP address and port for inclusion in the SDP of the INVITE. Normally, TURNs would be used in conjunction with ICE when applied to SIP. For simplicities sake, TURN is showed without ICE.

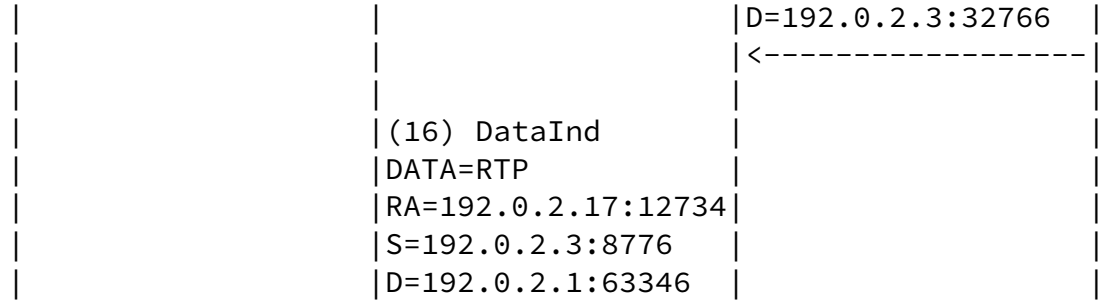
The client communicates with a SIP user agent on the public network. This user agent uses a 192.0.2.17:12734 for receipt of its RTP packets.

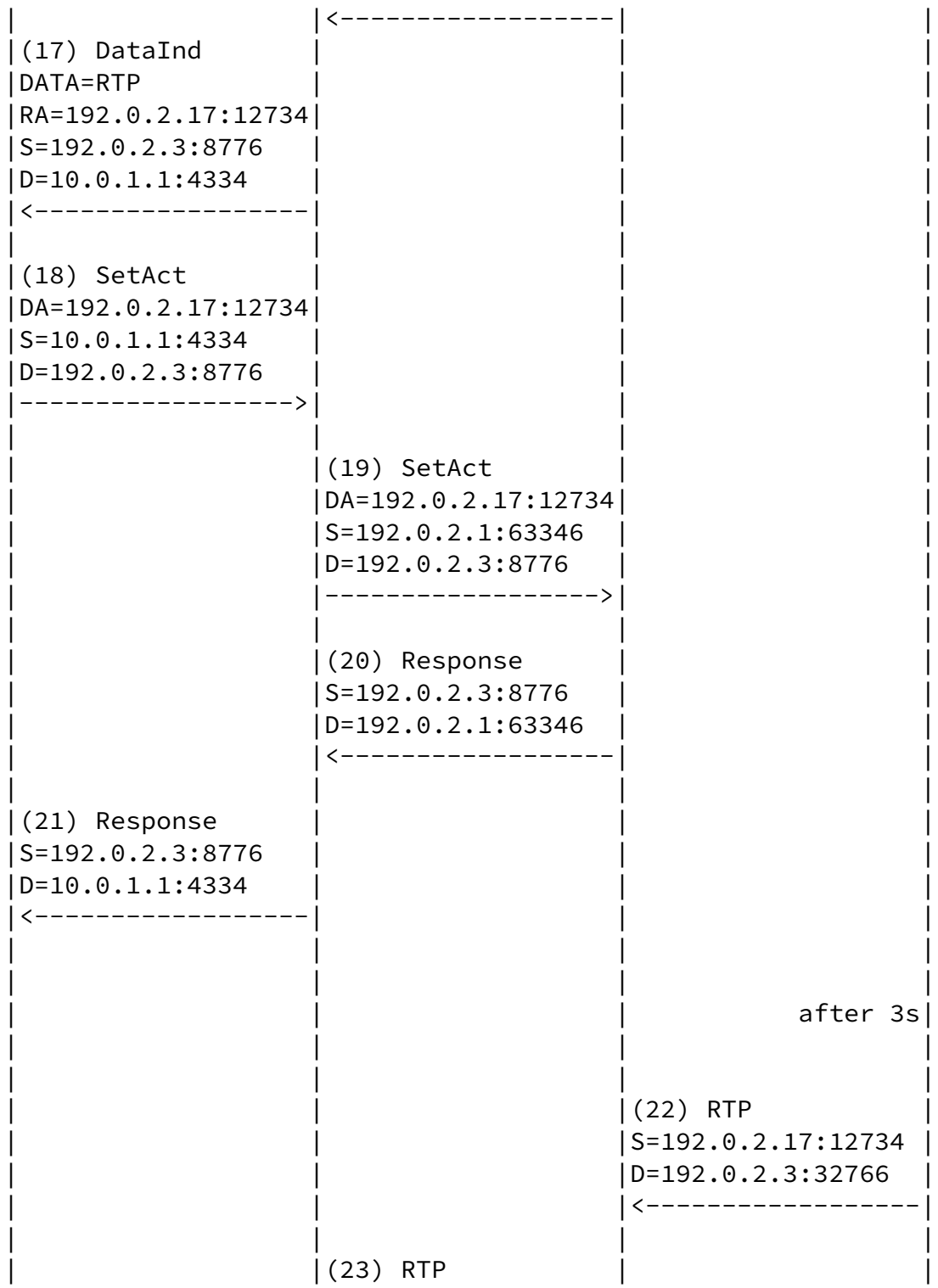



(3) Error
S=192.0.2.3:8776
D=192.0.2.1:63346
(4) Error
S=192.0.2.3:8776
D=10.0.1.1:4334
(5) Allocate
S=10.0.1.1:4334
D=192.0.2.3:8776
(6) Allocate
S=192.0.2.1:63346
D=192.0.2.3:8776
(7) Response
RA=192.0.2.3:32766
MA=192.0.2.1:63346
S=192.0.2.3:8776
D=192.0.2.1:63346


(8) Response
RA=192.0.2.3:32766
MA=192.0.2.1:63346
S=192.0.2.3:8776
D=10.0.1.1:4334
(9) INVITE
SDP=192.0.2.3:32766
(10) 200 OK


SDP=192.0.2.17:12734
(11) ACK
(12) Send
DATA=RTP
DA=192.0.2.17:12734
S=10.0.1.1:4334
D=192.0.2.3:8776
(13) Send
DATA=RTP
DA=192.0.2.17:12734
S=192.0.2.1:63346
D=192.0.2.3:8776
(14) RTP
S=192.0.2.3:32766
D=192.0.2.17:12734
Permission
Created
192.0.2.17
(15) RTP
S=192.0.2.17:12734


D=192.0.2.3:32766
(16) DataInd
DATA=RTP
RA=192.0.2.17:12734
S=192.0.2.3:8776
D=192.0.2.1:63346


|<------------------|
|(17) DataInd |
|DATA=RTP |
|RA=192.0.2.17:12734|
|S=192.0.2.3:8776 |
|D=10.0.1.1:4334 |
|<------------------|
|(18) SetAct |
|DA=192.0.2.17:12734|
|S=10.0.1.1:4334 |
|D=192.0.2.3:8776 |
|------------------>|
|(19) SetAct |
|DA=192.0.2.17:12734|
|S=192.0.2.1:63346 |
|D=192.0.2.3:8776 |
|------------------>|
|(20) Response |
|S=192.0.2.3:8776 |
|D=192.0.2.1:63346 |
|<------------------|
|(21) Response |
|S=192.0.2.3:8776 |
|D=10.0.1.1:4334 |
|<------------------|
after 3s
|(22) RTP |
|S=192.0.2.17:12734 |
|D=192.0.2.3:32766 |
|<------------------|
|(23) RTP |

|S=192.0.2.3:8776 |

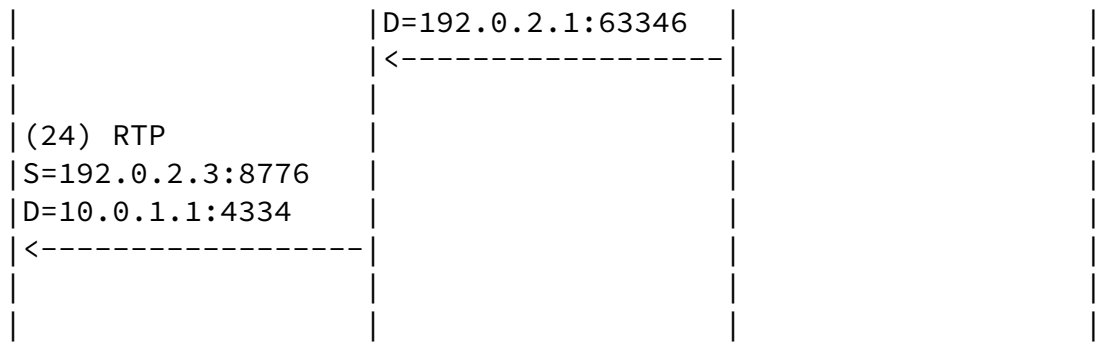


Figure 14

The call flow is shown in Figure 14. The client allocates a port from the local operating system on its private interface, obtaining 4334. It then attempts to secure a port for RTP traffic. RTCP processing is not shown. The client sends an Allocate request (1) with a source address (denoted by S) of 10.0.1.1:4334 and a destination (denoted by D) of 192.0.2.3:8776. This passes through the NAT (2), which creates a mapping from the 192.0.2.1:63346 to the source IP address and port of the request, 10.0.1.1:4334. This request is received at the STUN server, which challenges it (3), requesting credentials. This response is passed to the client (4). The client retries the request, this time with credentials (5). This arrives at the server (6). The request is now authenticated. The server provides a UDP allocation, 192.0.2.3:32766, and places it into the RELAY-ADDRESS (denoted by RA) in the response (7). It also reflects the source IP address and port of the request into the MAPPED-ADDRESS (denoted by MA) in the response. This passes through the NAT to the client (8). The client now proceeds to perform a basic SIP call setup. In message 9, it includes the relay address into the SDP of its INVITE. The called party responds with a 200 OK, and includes its IP address - 192.0.2.17:12734. The exchange completes with an ACK (11).

Next, user A sends an RTP packet. Since the active destination has not been set, the client decides to use the Send indication. It does so, including the RTP packet as the contents of the DATA attribute. The REMOTE-ADDRESS attribute (denoted by DA) is set to 192.0.2.17: 12734, learned from the 200 OK. This is sent through the NAT (message 12) and arrives at the STUN server (message 13). The server extracts the data contents, and sends the packet towards REMOTE-ADDRESS (message 14). Note how the source address and port in this packet is 192.0.2.3:32766, the allocated transport address given to the client. The act of sending the packet with Send causes the STUN server to install a permission for 192.0.2.17.

Indeed, the called party now sends an RTP packet toward the client

(message 15). This arrives at the STUN server. Since a permission has been set for the IP address in the source of this packet, it is accepted. As no active destination is set, the STUN server encapsulates the contents of the packet in a Data Indication (message 16), and sends it towards the client. The REMOTE-ADDRESS attribute (denoted by RA) indicates the source of the packet - 192.0.2.17: 12734. This is forwarded through the NAT to the client (message 17).

The client decides to optimize the path for packets to and from 192.0.2.17:12734. So, it issues a Set Active Destination request (message 18) with a REMOTE-ADDRESS of 192.0.2.17:12734. This passes through the NAT and arrives at the STUN server (message 19). This generates a successful response (message 20) which is passed to the client (message 21). At this point, the server and client are in the transitioning state. A little over 3 seconds later (by default), the state machines transition back to "Set". Until this point, packets from the called party would have been relayed back to the client in Data Indications. Now, the next RTP packet shows up at the STUN server (message 22). Since the source IP address and port match the active destination, the RTP packet is relayed towards the client without encapsulation (message 23 and 24).

16. Acknowledgements

The authors would like to thank Marc Petit-Huguenin for his comments and suggestions.

17. References

17.1. Normative References

[1] Rosenberg, J., "Session Traversal Utilities for (NAT) (STUN)", draft-ietf-behave-rfc3489bis-06 (work in progress), March 2007.

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

17.2. Informative References

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

[4] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with

Session Description Protocol (SDP)", RFC 3264, June 2002.

[5] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[6] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[7] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[8] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-16 (work in progress), June 2007.

[9] Audet, F. and C. Jennings, "NAT Behavioral Requirements for Unicast UDP", draft-ietf-behave-nat-udp-08 (work in progress), October 2006.

Authors' Addresses

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
Email: jdrosen@cisco.com
URI: http://www.jdrosen.net

Rohan Mahy
Plantronics

Email: rohan@ekabal.com

Christian Huitema

Microsoft
One Microsoft Way
Redmond, WA 98052-6399
US

Email: huitema@microsoft.com

Full Copyright Statement

Intellectual Property

specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).