TOC

Behave	J. Rosenberg
Internet-Draft	Cisco
Intended status: Standards Track	R. Mahy
Expires: May 20, 2008	Plantronics
	P. Matthews
	Avaya
	D. Wing
	Cisco
	November 17, 2007

Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)
draft-ietf-behave-turn-05

Status of this Memo

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 hosts behind address dependent NATs. The extension purposefully restricts the ways in which the relayed address can be used. In particular, it prevents users from running general purpose servers on ports obtained from the TURN server.

Table of Contents

1. Introduction

TOC

Session Traversal Utilities for NAT (STUN) [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.) provides a suite of tools for facilitating the traversal of NAT. Specifically, it defines the Binding method, 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 [RFC4787] (Audet, F. and C. Jennings, “Network Address Translation (NAT) Behavioral Requirements for Unicast UDP,” January 2007.) 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 UDP 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 transport addresses are from IP addresses belonging to 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 to STUN, called TURN, that allows a client to request an address on the TURN server, so that the TURN server acts as a relay. This extension defines a handful of new STUN methods. The Allocate method 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 TURN server. A transport address which relays through an intermediary is called a relayed transport address.

Though a relayed transport 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 (Rosenberg, J. and H. Schulzrinne, “An Offer/Answer Model with Session Description Protocol (SDP),” June 2002.) [RFC3264], is Interactive Connectivity Establishment (ICE) [I-D.ietf-mmusic-ice] (Rosenberg, J., “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” October 2007.). Though originally invented for Voice over IP applications, TURN is designed to be a general-purpose relay mechanism for NAT traversal.

2. Terminology

TOC

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 (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.) [RFC2119].

Relayed Transport Address: A transport address that terminates on a server, and is forwarded towards the client. The TURN 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(s).

Peer: A node with which the TURN client wishes to communicate. The TURN server relays traffic between the TURN client and its peer(s).

Allocation: The IP address and port granted to a client through an Allocate request, along with related state, such as permissions and expiration timers.

5-tuple: A combination of the source IP address and port, destination IP address and port, and transport protocol (UDP, or TCP). It uniquely identifies a TCP connection 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 remote peers that match an existing permission.

3. Overview of Operation

TOC

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. Elsewhere in the Internet are one or more peers that the TURN client wishes to communicate with. This specification defines a framing mechanism and several new STUN methods. Together, these add the ability for a STUN server to act as a packet relay.

The framing mechanism serves two purposes. First, it contains a length field that allow TURN nodes to find the boundaries between chunks of application data when the communication with the TURN server is over a stream-based transport such as TCP. Second, it carries a channel number. Channel zero is used for TURN control messages, while the other channel numbers are used for application data traveling to or from various peers. The channel number allows the client to know which peer sent data to it, and to specify which peer is to be the recipient of data. Application data flowing on any non-zero channel is unencapsulated, meaning that the application data starts immediately after the framing header. The framing header is just four bytes. This allows TURN to operate with minimal overhead, which is important for the real-time protocols it is designed to support. Application data can also flow in encapsulated format, meaning that it is carried in certain TURN messages on channel 0. Channel numbers are independent in each direction: for example, channel 5 might indicate one peer in the client to server direction, but a different peer in the server to client direction.

When the client wants to obtain a relayed transport address, the client first 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 transaction just reserves a transport address on the TURN server. Data does not flow through an allocated transport address until the TURN client asks the TURN server to open a permission, which is done with a Send Indication. 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 encapsulated in a Data Indication. Since multiple permissions can be open simultaneously, the Data Indication contains the PEER-ADDRESS attribute so the TURN client knows which peer sent the data, and a CHANNEL-NUMBER attribute so the client knows how the server will refer to traffic from this peer when sent unencapsulated. Likewise when the client initially sends to a new peer, it uses a Send Indication with the peer address in the PEER-

ADDRESS attribute, along with a channel number so the server knows how the client will refer to unencapsulated data to this peer.

```
     TURN                   TURN          peer
     client                 server
       |--- Allocate Req  -->|             |
       |<-- Allocate Resp ---|             |
       |                     |             |
       |--- Send (chan 2) -->|    data     |
       |                     |============>|
       |<-- ChannelConfirm --|             |
       |                     |    data     |
       |                     |<============|
       |<-- Data (chan 5) ---|             |
       |--- ChannelConfirm ->|             |
       |                     |             |
       |--- [2] + data ----->|    data     |
       |                     |============>|
       |                     |    data     |
       |                     |<============|
       |<-- [5] + data ------|             |
```

Figure 1: Example Usage of Channels

When the client and server communicate over UDP, data and control messages can arrive out of order. For this reason, the client needs to verify the server knows the client channel mapping before the client sends unencapsulated, and the server needs to verify the client knows the server channel mapping before the server sends unencapsulated. When the client and server communicate over UDP, a Channel Confirmation indication is sent after the Send (or Data) indication so the client (or server) knows that it can send unencapsulated.
Figure 1 (Example Usage of Channels) demonstrates how this works. The client performs an Allocate Request, and gets a response. It decides to send data to a specific peer. Initially, it sends data to that peer using a TURN Send indication on channel 0. That Send Indication tells the TURN server that, once confirmed, the client will send data unencapsulated to that peer on channel 2. Whenever the TURN server receives a Send indication, it stores the mapping from channel number to peer, and sends a ChannelConfirm indication (on channel 0). Once the confirmation has been received by the client, the client can send data to the peer on channel 2. Prior to receipt of the ChannelConfirm, any other data the client wishes to send to the peer is sent using Send indications, all of which indicate that channel 2 is to be used for unencapsulated data. The same procedure happens from server to client;

the TURN server initially sends data using a Data indication on channel 0, and once confirmed with a ChannelConfirm, it can send it unencapsulated on its selected channel (channel 5 in the example). Over a reliable transport, such as TCP, the confirmation step is not needed so the Channel Confirmation indication is not used. Clients can immediately send the next piece of data to the peer on the requested channel.
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.

3.1. Transports

TOC

TURN clients can communicate with a TURN server using UDP, TCP, or TLS over TCP. A TURN server can then relay traffic between a reliable transport used between the client and server (TCP or TLS over TCP), and UDP used from server to peer. 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 TURN framing or the length of the DATA attribute in a Send Indication. Likewise, when a UDP datagram is received by the TURN server and relayed to the client over a stream-based transport, the length of the datagram is the length of the TURN framing or Data Indication's DATA attribute.
The following table shows the possible combinations of transport protocols from client to server and from server to peer:

client to TURN server	TURN server to peer
UDP	UDP
TCP	UDP
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 when it would not be able to communicate with the same server 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.
In addition, an extension to TURN is planned to add support for TCP allocations [I-D.ietf-behave-turn-tcp] (Perreault, S. and J. Rosenberg, “Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations,” March 2010.).

3.2. About Tuples

TOC

To relay data to and from the correct location, the TURN server maintains an association between the 5-tuple used to communicate with the client and the 5-tuple used to communicate with each of the client's peers. The 5-tuple on the client side will consist of the client's reflexive address -- the apparent source address and port of the client (typically as rewritten by the last NAT)--and the destination address and port used by the TURN server. The figure below shows a typical topology. In this diagram, the client 5-tuple is for a UDP flow between 192.0.2.1:7000 and 192.0.2.15:3490. The 5-tuple between the TURN server and Peer B is for a UDP flow between 192.0.2.15:9000 (the TURN allocated address) and 192.0.2.210:18200.

> 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.

```
                                                            +---------+
                                                            |         |
                                                            |         |
                                                          / |  Peer A |
      Client's              TURN                        //  |         |
      Host                  Server                     /    |         |
      Address               Address                  //     +---------+
   10.1.1.2:17240       192.0.2.15:3490             / 192.0.2.180:16400
         |                     |                   //
         |        +-+          |                  /
         |        | |          |                 /
         v        | |          |               //      192.0.2.210:18200
   +---------+    | |          |+---------+   /              +---------+
   |         |    |N|          ||         | //               |         |
   | TURN    |    | |          v|  TURN   |/                 |         |
   | Client  |----|A|----------|  Server |------------------|  Peer B |
   |         |    | |^         |         |^                ^|         |
   |         |    |T||         |         ||                ||         |
   +---------+    | ||         +---------+|                |+---------+
                  | ||                    |                |
                  | ||                    |                |
                  +-+|                    |                |
                     |                    |                |
                     |
                   Client's           TURN              Peer B
                   Reflexive          Allocated         Transport
                   Address            Address           Address
                192.0.2.1:7000    192.0.2.15:9000     192.0.2.210:18200
```

Figure 2

3.3. Keepalives

Since the main purpose of STUN and TURN is to traverse NATs, it is natural to consider which elements are responsible for generating sufficient periodic traffic to insure that NAT bindings stay alive. TURN clients need to send data frequently enough to keep both NAT bindings and the TURN server permissions fresh. Like NAT bindings, the TURN server permissions are refreshed by ordinary data traffic relayed from the client to the peer. Unlike permissions, allocations on the TURN server have an explicit expiration time and need to be refreshed explicitly by the client with a TURN Refresh request. When an allocation expires, all permissions associated with that allocation are automatically deleted.

4. TURN Framing Mechanism

TOC

All TURN control messages and all application data sent between the client and the server MUST start with the TURN framing header. This header is used for two purposes: indicating the channel number, and for framing.
TURN uses a channel number to distinguish control traffic from data, and to distinguish among multiple peers using the same allocation. Channel number zero is reserved for TURN control messages. All TURN requests, responses and indications between the client and server MUST be sent on channel 0, and MUST NOT be sent on any other channel. Channel 0xFFFF is reserved for future use and MUST NOT be used by clients or servers compliant to this specification. Other channel numbers are assigned and communicated as described in Section 7 (Sending and Receiving Data). Because the framing is always used, TURN needs to run on a separate port number from unframed STUN requests. Over stream-based transports, the TURN client and server also need to include an explicit length so that the TURN server can perform conversion from streams to datagrams and vice versa. TURN framing has a 2 octet channel number and a 2 octet length field. Over stream-based transports, the length field counts the number of octets immediately after the length field itself. Over UDP the length is always set to zero.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Channel Number        |            Length             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Channel numbers are always defined within a particular allocation. If a client has multiple allocations on a TURN server, there is no relationship whatsoever between the channel numbers in each allocation. Once created, a channel number persists for the lifetime of the allocation. There is no way to explicitly remove a channel. Consequently, a client which obtains an allocation with the intent of holding it for extremely long periods, possibly for communication with many different peers over time, may eventually exhaust the set of channels. In that case, the client will need to obtain a new allocation.

TOC

5. General Behavior

After the initial Allocate transaction, all subsequent TURN transactions need to be sent in the context of a valid allocation. The source and destination IP address and ports for these TURN messages MUST match the internal 5-tuple of an existing allocation. These are processed using the general server procedures in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.) with a few important additions. For requests (in this specification, the only subsequent request possible is a Refresh request), if there is no matching allocation, the server MUST generate a 437 (Allocation Mismatch) error response. For indications, if there is no matching allocation, the indication is silently discarded. An Allocate request MUST NOT be sent by a client within the context of an existing allocation. Such a request MUST be rejected by the server with a 437 (Allocation Mismatch) error response.

A subsequent request MUST be authenticated using the same username and realm as the one used in the Allocate request that created the allocation. If the request was authenticated but not with the matching credential, the server MUST reject the request with a 401 (Unauthorized) error response.

When a server returns an error response, it MAY include an ALTERNATE-SERVER attribute if it has positive knowledge that the problem reported in the error response will not be a problem on the alternate server. For example, a 443 response (Invalid IP Address) with an ALTERNATE-SERVER means that the other server is responsible for that IP address. A 442 (Unsupported Transport Protocol) with this attribute means that the other server is known to support that transport protocol. A 507 (Insufficient Capacity) means that the other server is known to have sufficient capacity. Using the ALTERNATE-SERVER mechanism in the 507 (Insufficient Capacity) response can only be done if the rejecting server has definitive knowledge of available capacity on the target. This will require some kind of state sharing mechanism between TURN servers, which is beyond the scope of this specification. If a TURN server attempts to redirect to another server without knowledge of available capacity, it is possible that all servers are in a congested state, resulting in series of rejections that only serve to further increase the load on the system. This can cause congestion collapse.

If a client sends a request to a server and gets a 500 class error response without an ALTERNATE-SERVER, or the transaction times out without a response, and the client was utilizing the SRV procedures of [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.) to contact the server, the client SHOULD try another server based on those procedures. However, the client SHOULD cache the fact that the request to this server failed, and not retry that server again for a configurable period of time. Five minutes is RECOMMENDED.

TURN clients and servers MUST NOT include the FINGERPRINT attribute in any of the methods defined in this document.

6. Managing Allocations

TOC

Communications between a TURN client and a TURN server on a new flow begin with an Allocate transaction. All subsequent transactions happen in the context of that allocation. The client refreshes allocations and deallocates them using a Refresh transaction.

6.1. Client Behavior

TOC

6.1.1. Initial Allocate Requests

TOC

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 [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.). Clients MUST implement the long term credential mechanism defined in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.), and be prepared for the server to use it.
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.

> OPEN ISSUE: Bandwidth is very much underspecified. Is anyone actually using it for capacity planning? If not we should remove.

The client MAY request a particular lifetime for the allocation by including it in the LIFETIME attribute in the request.
The client MUST include a REQUESTED-TRANSPORT attribute. In this specification, the REQUESTED-TRANSPORT will always be UDP. This attribute is included to allow for future extensions to TURN.
The client MAY include a REQUESTED-PORT-PROPS 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) [RFC3550] (Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, “RTP: A

Transport Protocol for Real-Time Applications," July 2003.) requires that RTP and RTCP ports be an adjacent 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.

Processing of the response follows the general procedures of [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for (NAT) (STUN)," July 2008.). 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 value of the LIFETIME attribute in the response indicates the amount of time after which the server will expire the allocation, if not refreshed with a Refresh 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 UDP 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.

6.1.2. Refresh Requests

TOC

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 a Refresh transaction if it wishes to keep the allocation.

To perform a refresh, the client generates a Refresh Request. The client MUST use the same username, realm and password for the Refresh request as it used in its initial Allocate Request. The Refresh request MAY contain a proposed LIFETIME attribute. The client MAY include a BANDWIDTH attribute if it wishes to request more or less bandwidth than in the original request. If absent, it indicates no change in the requested bandwidth from the Allocate request. The client MUST NOT include a REQUESTED-IP, REQUESTED-TRANSPORT, or REQUESTED-PORT-PROPS attribute in the Refresh request.

In a successful response, the LIFETIME attribute indicates the amount of additional time (the number of seconds after the response is received) that the allocation will live without being refreshed. A successful response will also contain a BANDWIDTH attribute, indicating the bandwidth the server is allowing for this allocation. Note that an

error response does not imply that the allocation has expired, just that the refresh has failed.
If a client no longer needs an allocation, it SHOULD perform an explicit deallocation. If the client wishes to explicitly remove the allocation because it no longer needs it, it sends a Refresh request, but sets the LIFETIME attribute to zero. This will cause the server to remove the allocation, and all associated permissions and channel numbers. 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.

6.2. Server Behavior

TOC

The server first processes the request according to the base protocol procedures in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.), extended with the procedures for the long-term credential mechanism.

6.2.1. Initial Allocate Requests

TOC

When the server receives an Allocate request, the server attempts to allocate a relayed transport address. It first looks for the BANDWIDTH attribute in 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.

6.2.1.1. REQUESTED-TRANSPORT

TOC

First, the server checks for the REQUESTED-TRANSPORT attribute. This indicates the transport protocol requested by the client. This specification defines a value for UDP only, but support for TCP allocations is planned in [I-D.ietf-behave-turn-tcp] (Perreault, S. and J. Rosenberg, “Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations,” March 2010.).

> 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. If the request did not contain a REQUESTED-TRANSPORT attribute, the server MUST use the same transport protocol as the request arrived on.

6.2.1.2. REQUESTED-IP

TOC

Next, the server checks for the REQUESTED-IP attribute. If present, it indicates a specific IP address from which the client would like its transport address allocated. (The client could do this if it requesting the second address in a specific port pair). If this IP address 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.

6.2.1.3. REQUESTED-PORT-PROPS

TOC

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 specific port is not available (in use or reserved), the server MUST reject the request with a 444 (Invalid Port) response. 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 predate RFC 3550. These implementations use the port numbering conventions in (now obsolete) RFC 1889.

6.2.1.4. Creating the Allocation

TOC

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.

Once a port is allocated, the server associates the allocation with the 5-tuple used to communicate between the client and the server. For TCP, this amounts to associating the TCP connection from the TURN client with the allocated transport address.

The new allocation MUST also be associated with the username, password and realm used to authenticate the request. These credentials are used in all subsequent requests to ensure that only the same client can use or modify the allocation it was given.

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 duration at its discretion. Ten minutes is RECOMMENDED. 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 7.2.3 (Allocation Activity Timer and Permission Timeout) discusses behavior when the timer fires. Note that the LIFETIME attribute an Allocate request can be zero, though this is effectively a no-op, since it will create and destroy the allocation in one transaction.

6.2.1.5. Sending the Allocate Response

TOC

Once the port has been obtained and the allocation expiration timer has been started, the server generates an Allocate Response using the general procedures defined in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for (NAT) (STUN)," July 2008.), including the ones for long term authentication. 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-allocation limit that the server is placing on the bandwidth usage on each binding. Such limits are needed to prevent against denial-of-service attacks (See Section 11 (Security Considerations)).

TOC

6.2.2. Refresh Requests

A Refresh request is processed using the general server and long term authentication procedures in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.). It is used to refresh and extend an allocation, or to cause an immediate deallocation. It is processed as follows.

First, the request MUST be authenticated using the same shared secret as the one associated with the allocation. If the request was authenticated but not with such a matching credential, the server MUST generate a Refresh Error Response with a 401 response.

If the Refresh request contains a BANDWIDTH attribute, the server checks that it can relay the requested volume of traffic.

Finally, a Refresh Request will set a new allocation expiration timer for the allocation, effectively canceling the previous allocation expiration timer. As with an Allocate request, the server can offer a shorter allocation lifetime, but never a longer one.

A success Refresh response MUST contain a LIFETIME attribute and a BANDWIDTH attribute.

---

7. Sending and Receiving Data

TOC

As described in Section 4 (TURN Framing Mechanism), TURN allows a client to send and receive data without utilizing TURN Send and Data indications, by sending and receiving them on channels. Before sending client-to-peer or peer-to-client data for a new peer, a TURN client or server needs to assign a channel number that corresponds to that remote peer. Once a channel number is assigned, it remains assigned through the duration of the allocation. It cannot be unassigned or reassigned to a different peer.

7.1. Client Behavior

TOC

TOC

7.1.1. Sending

When the client wants to forward data to a peer, it checks if it has assigned a channel number for communications with this peer (as identified by its IP address and port) over this allocation:

- If one has not been assigned, the client assigns one of its own choosing. This channel number MUST be one that is currently unassigned by the client for this allocation. It MUST be between 1 and 65534. It is RECOMMENDED that the client choose one of the unassigned numbers randomly, rather than sequentially. The state of the channel is set to unconfirmed.

- If one has been assigned, that channel MUST be selected.

Next, the client checks if the channel number has been confirmed by the server. If the channel number has been confirmed, the client simply sends the data to the TURN server with the appropriate channel number in the TURN framing.
If the channel number has not been confirmed, the client creates a Send indication. It places the selected channel number in a CHANNEL-NUMBER attribute, the peer IP address and port in a PEER-ADDRESS attribute, and puts the data to be sent in a DATA attribute. (If the client just wishes to create a permission, it can omit the DATA attribute.) If the Send indication is sent over a reliable transport (ex: TCP), the client marks that the channel number as confirmed. When the client receives a ChannelConfirmation Indication, and the channel number, IP address and port match the channel number assigned to that peer, the client marks that the channel number is confirmed.
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. A ChannelConfirmation Indication just means that some Send indication was received by the TURN server. It does not mean that a specific Send indication 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.

7.1.2. Receiving

TOC

When the client receives a Data indication, it:

- records the channel number used by the server (from the CHANNEL-NUMBER attribute) and associates it with the IP address and port in the PEER-ADDRESS attribute, which identify the peer that sent the data. The resulting mapping from channel number to transport address MUST be stored by the client for the duration of the allocation.

- delivers the contents of the DATA attribute to the client application as if it was received from the peer's IP address and port.

- If the Data indication was received over UDP, the client MUST confirm the channel used by the server, by sending a ChannelConfirmation Indication to the server. This indication MUST contain the same PEER-ADDRESS and CHANNEL-NUMBER attributes included in the Data indication. This indication is sent to the server on channel 0 using the 5-tuple associated with this allocation. Note that, due to round trip delays, a client may receive several Data indications with the same channel number for the same remote peer. It MUST process each as defined here, resulting in several ChannelConfirmation indications.

When the client receives unencapsulated data, it checks the received channel number. If the client has a mapping associated with the server channel number it delivers the data to the client application as if it was received directly from that peer. Otherwise, it silently discards the data.

7.2. Server Behavior

TOC

7.2.1. Receiving Data from the Client

TOC

When the server receives a Data indication from the client, it:

- records the channel number used by the client (from the CHANNEL-NUMBER attribute) and associates it with the IP address and port in the PEER-ADDRESS attribute, which identify the peer to which the data is to be sent. The resulting mapping from channel number to peer transport address MUST be stored by the server for the duration of the allocation.

- sends the contents of the DATA attribute in a UDP datagram, sending it to the PEER-ADDRESS and sending from the allocated transport address.

- if one doesn't exist, creates a permission for the IP address from the PEER-ADDRESS (the port is ignored), and attaches the permission to the allocation

- checks if a timer has been set for this permission. If none has been started, the server starts one. It is RECOMMENDED that it have a value of sixty seconds. If the timer is already running, it MUST be reset.

- If the Send indication was received over UDP, the server MUST confirm the channel used by the client, by sending a ChannelConfirmation Indication to the client. This indication MUST contain the same PEER-ADDRESS and CHANNEL-NUMBER attributes included in the Send indication. This indication is sent to the client on channel 0 using the 5-tuple associated with this allocation. Note that, due to round trip delays, a server may receive several Send indications with the same channel number for the same remote peer. It MUST process each as defined here, resulting in several ChannelConfirmation indications.

When the server receives unencapsulated data, it checks the received channel number:

- If the server has a mapping associated with the client channel number it:

  - sends a UDP datagram to the peer using the transport address from the mapping, and sends from the allocated transport address.

  - checks if a permission activity timer is running for the destination IP address of the peer. If one is not running, the server starts one. It is RECOMMENDED that it have a value of sixty seconds. If the timer is already running, it MUST be reset.

- If the server has no mapping, it silently discards the data.

7.2.2. Receiving Data from Peers

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 is not used), the UDP packet is accepted. Otherwise, it is discarded. If the packet is accepted, it is forwarded to the client as described below.

The server checks if it has assigned a channel number for communications from this peer (as identified by its IP address and port) over this allocation:

* If one has not been assigned, the client assigns one of its own choosing. This channel number MUST be one that is currently unassigned by the server for this allocation. It MUST be between 1 and 65534. It is RECOMMENDED that the server choose one of the unassigned numbers randomly, rather than sequentially. The state of the channel is set to unconfirmed.

* If one has been assigned, that channel MUST be selected.

Note that data from peers does not reset the permission activity timer. Next, the server checks if the channel number has been confirmed by the client. If the channel number has been confirmed, the server simply sends the data to the client with the appropriate channel number in the TURN framing.
If the channel number has not been confirmed, the server creates a Data indication. It places the selected channel number in a CHANNEL-NUMBER attribute, the peer IP address and port in a PEER-ADDRESS attribute, and puts the data to be sent in a DATA attribute. If the Data indication is sent over a reliable transport (ex: TCP), the server marks that the channel number as confirmed. When the server receives a ChannelConfirmation Indication, and the channel number, IP address and port match the channel number assigned to that peer, the server marks that the channel number is confirmed.
Since Data is an Indication, it generates no response. The server does not provide reliability for the data. When sending over a reliable transport to the client, if the server is unable to send the data received from the peer (for example, because the TCP connection cannot accept any more messages right now), it can silently discards UDP data received from 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.

7.2.3. Allocation Activity Timer and Permission Timeout

TOC

When the allocation activity timer expires, the server MUST destroy the allocation. This involves freeing the allocated transport address, deleting permissions and channel numbers, and removing other state associated with the allocation.

When a permission times out, the TURN server MUST NOT forward a packet from that TURN peer to the TURN client.

8. New Attributes

TOC

This STUN extension defines the following new attributes:

```
0x000C: CHANNEL-NUMBER
0x000D: LIFETIME
0x0010: BANDWIDTH
0x0012: PEER-ADDRESS
0x0013: DATA
0x0016: RELAY-ADDRESS
0x0018: REQUESTED-PORT-PROPS
0x0019: REQUESTED-TRANSPORT
0x0022: REQUESTED-IP
```

8.1. CHANNEL-NUMBER

TOC

The channel number attribute represents the channel number assigned by the sender, that corresponds with the peer specified in the PEER-ADDRESS attribute. It is a 16-bit unsigned integer, plus two octets of padding which MUST be set to zero.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Channel Number         |         Reserved = 0          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.2. LIFETIME

TOC

The lifetime attribute represents the duration for which the server will maintain an allocation in the absence of a refresh. It is a 32 bit unsigned integral value representing the number of seconds remaining until expiration.

TOC

8.3. BANDWIDTH

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

8.4. PEER-ADDRESS

TOC

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

8.5. DATA

TOC

The DATA attribute is present in most 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).

8.6. RELAY-ADDRESS

TOC

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 XOR-MAPPED-ADDRESS.

8.7. REQUESTED-PORT-PROPS

TOC

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:

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |       Reserved = 0        | A |    Specific Port Number       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 value of the A field is 00 (no specific port alignment), then the Specific Port Number field can either be 0 or some non-zero port number. If the Specific Port Number field is 0, then the client is not putting any restrictions on the port number it would like allocated. If the Specific Port Number is some non-zero port number, then the client is requesting that the server allocate the specified port.
If the value of the A field is 01 (odd port number), then the Specific Port Number field must be zero, and the client is requesting the server allocate an odd-numbered port.
If the value of the A field is 10 (even port number), then the Specific Port number field must be zero, and the client is requesting the server allocate an even-numbered port.
If the value of the A field is 11 (even port number; reserve next higher port), then the Specific Port Number field must be zero, and the client is requesting the server allocate an even-numbered port. In addition, the client is requesting the server reserve the next higher port (i.e., N+1 if the server allocates port N), and should only allocate the N+1 port number if it is explicit requested (with a subsequent request specifying that exact port number)
In all cases, if a port with the requested properties cannot be allocated, the server responds with a error response with an error code of 444 (Invalid Port).

8.8. 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:	Reserved for 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 allocation, the server will relay data between the two transports. Extensions to TURN can define additional transport protocols in an IETF-consensus RFC.

8.9. REQUESTED-IP

TOC

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 XOR-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.

---

9. New Error Response Codes

TOC

This document defines the following new Error response codes:

437 (Allocation Mismatch): A request was received by the server that requires an allocation to be in place, but there is none, or a request was received which requires no allocation, but there is one.

442 (Unsupported Transport Protocol): The Allocate request asked for a transport protocol to be allocated that is not supported by the server. If the server is aware of another server that supports the requested protocol, it SHOULD include the other server's address in an ALTERNATE-SERVER attribute in the error response.

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.

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.

TOC

10. Client Discovery of TURN Servers

The STUN extensions introduced by TURN differ from the binding requests defined in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.) in that they are sent with additional framing and demand substantial resources from the TURN server. In addition, it seems likely that administrators might want to block connections from clients to the TURN server for relaying separately from connections for the purposes of binding discovery. As a consequence, TURN runs on a separate port from STUN. The client discovers the address and port of the TURN server using the same DNS procedures defined in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.), but using an SRV service name of "turn" (or "turns" for TURN over TLS) instead of just "stun".
For example, to find TURN servers in the example.com domain, the TURN client performs a lookup for '_turn._udp.example.com', '_turn._tcp.example.com', and '_turns._tcp.example.com' if the STUN client wants to communicate with the TURN server using UDP, TCP, or TLS over TCP, respectively.

---

11. Security Considerations

TURN servers allocate bandwidth and port resources to clients, in contrast to the Binding method defined in [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.). Therefore, a TURN server requires authentication and authorization of STUN requests. This authentication is provided by mechanisms defined in the STUN specification itself, in particular digest authentication.
Because TURN servers allocate resources, they can be susceptible to denial-of-service attacks. All Allocate transactions 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 limit 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 such 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 [I-D.ietf-behave-rfc3489bis] (Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” July 2008.) [[TODO: Update section number references to 3489bis]]. 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 explicitly authorize permission to receive data from a peer, one IP address at a time. Thus, it does not provide a general technique for externalizing 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 transactions does not appear to be that important. If required, it can be provided by running TURN over TLS.

TURN does not and cannot guarantee that UDP data is delivered in sequence or to the correct address. As most TURN clients will only communicate with a single peer, the use of a single channel number will be very common. Consider an enterprise where Alice and Bob are involved in separate calls through the enterprise NAT to their corporate TURN server. If the corporate NAT reboots, it is possible that Bob will obtain the exact NAT binding originally used by Alice. If Alice and Bob were using identical channel numbers, Bob will receive unencapsulated data intended for Alice and will send data accidentally to Alice's peer. This is not a problem with TURN. This is precisely what would happen if there was no TURN server and Bob and Alice instead provided a (STUN) reflexive transport address to their peers. If detecting this misdelivery is a problem, the client and its peer need to use message integrity on their data.

One TURN-specific DoS attack bears extra discussion. An attacker who can corrupt, drop, or cause the loss of a Send or Data indication sent over UDP, and then forge a Channel Confirmation indication for the corresponding channel number, can cause a TURN client (server) to start sending unencapsulated data that the server (client) will discard. Since indications are not integrity protected, this attack is not prevented by cryptographic means. However, any attacker who can generate this level of network disruption could simply prevent a large fraction of the data from arriving at its destination, and therefore protecting against this attack does not seem important. The ChannelConfirmation forging attack is not possible when the client to server communication is over TCP or TLS over TCP.

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 TURN server, so that the user's IP addresses are never revealed.

Any relay addresses learned through an Allocate request will not operate properly with IPSec Authentication Header (AH) (Kent, S., “IP Authentication Header,” December 2005.) [RFC4302] in transport or tunnel mode. However, tunnel-mode IPSec ESP (Kent, S., “IP Encapsulating Security Payload (ESP),” December 2005.) [RFC4303] should still operate.

12. IANA Considerations

TOC

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

12.1. New STUN Methods

TOC

```
Request/Response Transactions
0x003  :  Allocate
0x004  :  Refresh

Indications
0x006  :  Send
0x007  :  Data
0x009  :  Channel Confirmation
```

12.2. New STUN Attributes

TOC

0x000C:	CHANNEL-NUMBER
0x000D:	LIFETIME
0x0010:	BANDWIDTH
0x0012:	PEER-ADDRESS
0x0013:	DATA
0x0016:	RELAY-ADDRESS
0x0018:	REQUESTED-PORT-PROPS
0x0019:	REQUESTED-TRANSPORT
0x0022:	REQUESTED-IP

12.3. New STUN Response Codes

TOC

437	Allocation Mismatch
442	Unsupported Transport Protocol
443	Invalid IP Address
444	Invalid Port
486	Allocation Quota Reached
507	Insufficient Capacity

13. IAB Considerations

TOC

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 (Daigle, L. and IAB, “IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation,” November 2002.) [RFC3424]. 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 (Rosenberg, J., “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” October 2007.) [I-D.ietf-mmusic-ice].

TOC

14. Example

In this example, a TURN client is behind a NAT. This TURN client is running SIP. The client has a private address of 10.0.1.1. The TURN 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, TURN would be used in conjunction with ICE when applied to SIP. However, to keep the example simple, TURN is shown 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.

```
10.0.1.1            192.0.2.1           192.0.2.3          192.0.2.17
 Client                NAT            TURN Server            Peer
    |                   |                   |                   |
    |(1) Allocate       |(2) Allocate       |                   |
    |S=10.0.1.1:4334    |S=192.0.2.1:63346  |                   |
    |D=192.0.2.3:8776   |D=192.0.2.3:8776   |                   |
    |------------------>|------------------>|                   |
    |                   |                   |                   |
    |(4) Error          |(3) Error          |                   |
    |S=192.0.2.3:8776   |S=192.0.2.3:8776   |                   |
    |D=10.0.1.1:4334    |D=192.0.2.1:63346  |                   |
    |<------------------|<------------------|                   |
    |                   |                   |                   |
    |(5) Allocate       |(6) Allocate       |                   |
    |S=10.0.1.1:4334    |S=192.0.2.1:63346  |                   |
    |D=192.0.2.3:8776   |D=192.0.2.3:8776   |                   |
    |------------------>|------------------>|                   |
    |                   |                   |                   |
    |                   |          (allocates port 32766)       |
    |                   |                   |                   |
    |                   |                   |                   |
    |(8) Response       |(7) Response       |                   |
    |RA=192.0.2.3:32766 |RA=192.0.2.3:32766 |                   |
    |MA=192.0.2.1:63346 |MA=192.0.2.1:63346 |                   |
    |S=192.0.2.3:8776   |S=192.0.2.3:8776   |                   |
    |D=10.0.1.1:4334    |D=192.0.2.1:63346  |                   |
    |<------------------|<------------------|                   |
    |                   |                   |                   |
    |(9) SIP INVITE     |                   |                   |
    |SDP=192.0.2.3:32766|                   |                   |
    |---------------------------------------------------------->|
    |                   |                   |                   |
    |(10) SIP 200 OK    |                   |                   |
    |SDP=192.0.2.17:12734                   |                   |
    |<----------------------------------------------------------|
    |                   |                   |                   |
    |                   |                   |(11) RTP           |
    |                   |                   |S=192.0.2.17:12734 |
    |                   |                   |D=192.0.2.3:32766  |
    |                   |                   |<------------------|
    |                   |                   |                   |
    |                   |    (no permission; packet dropped)    |
    |                   |                   |                   |
    |(12) SIP ACK       |                   |                   |
    |---------------------------------------------------------->|
    |                   |                   |                   |
    |(13) Send Indic.   |(14) Send Indic.   |                   |
    |TURN Channel=0     |TURN Channel=0     |                   |
    |STUN DATA=RTP      |STUN DATA=RTP      |                   |
    |CHANNEL-NUMER=77   |CHANNEL-NUMBER=77  |                   |
    |PA=192.0.2.17:12734|PA=192.0.2.17:12734|                   |
    |S=10.0.1.1:4334    |S=192.0.2.1:63346  |                   |
    |D=192.0.2.3:8776   |D=192.0.2.3:8776   |                   |
    |------------------>|------------------>|                   |
    |                   |                   |                   |
    |                   |            permission created         |
    |                   |                   |                   |
    |                   |                   |(15) RTP           |
    |                   |                   |S=192.0.2.3:32766  |
    |                   |                   |D=192.0.2.17:12734 |
    |                   |                   |------------------>|
    |                   |                   |                   |
    |(17) ChannelConf   |(16) ChannelConf   |                   |
    |TURN Channel=0     |TURN Channel=0     |                   |
    |CHANNEL-NUMBER=77  |CHANNEL-NUMBER=77  |                   |
    |PA=192.0.2.17:12734|PA=192.0.2.17:12734|                   |
    |S=192.0.2.3:8776   |S=192.0.2.3:8776   |                   |
    |D=10.0.1.1:4334    |D=192.0.2.1:63346  |                   |
    |<------------------|<------------------|                   |
    |                   |                   |                   |
    |(18) TURN Framed   |(19) TURN Framed   |                   |
    |TURN Channel=77    |TURN Channel=77    |(20) RTP           |
    |S=10.0.1.1:4334    |S=192.0.2.1:63346  |S=192.0.2.3:32766  |
    |D=192.0.2.3:8776   |D=192.0.2.3:8776   |D=192.0.2.17:12734 |
    |------------------>|------------------>|------------------>|
    |                   |                   |                   |
    |(23) Data Indic.   |(22) Data Indic.   |                   |
    |TURN Channel=0     |TURN Channel=0     |                   |
    |CHANNEL-NUMBER=33  |CHANNEL-NUMBER=33  |(21) RTP           |
    |S=192.0.2.3:8776   |S=192.0.2.3:8776   |S=192.0.2.17:12734 |
    |D=10.0.1.1:4334    |D=192.0.2.1:63346  |D=192.0.2.3:32766  |
    |<------------------|<------------------|<------------------|
    |                   |                   |                   |
    |(24) ChannelConf   |(25) ChannelConf   |                   |
    |TURN Channel=0     |TURN Channel=0     |                   |
    |CHANNEL-NUMBER=33  |CHANNEL-NUMBER=33  |                   |
    |S=10.0.0.1:4334    |S=192.0.2.3:8776   |                   |
    |D=192.0.2.3:8776   |D=192.0.2.3:8776   |                   |
    |------------------>|------------------>|                   |
    |                   |                   |                   |
    |(28) TURN Framed   |(27) TURN Framed   |                   |
    |TURN Channel=33    |TURN Channel=33    |(26) RTP           |
    |S=192.0.2.3:8776   |S=192.0.2.3:8776   |S=192.0.2.17:12734 |
    |D=10.0.1.1:4334    |D=192.0.2.1:63346  |D=192.0.2.3:32766  |
    |<------------------|<------------------|<------------------|
    |                   |                   |                   |
```

Figure 3

The message flow is shown in Figure 3. In step 1-2, the client allocates a UDP port from the local operating system on its private interface, obtaining 4334. It then attempts to obtain a port for RTP traffic. RTCP processing is not shown in the example.

In step 1, 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 allocates a new UDP port (63346) on the NAT's public interface (192.0.2.1), and creates an internal mapping between the internal address 10.0.1.1:4334 and that external address 192.0.2.1:63346. The NAT sends this request to the TURN server (3). The TURN server challenges the request, requesting credentials by sending a STUN error and including the NONCE and REALM attributes. Message 3 is relayed, by the NAT, to the TURN client (4). The client sends a new request (from the same UDP port), including its credentials (5, 6). The TURN server authenticates the request. The TURN server allocates a new UDP port on one of its interfaces, 192.0.2.3:32766. The TURN server puts 192.0.2.3:32766 into the RELAY-ADDRESS (denoted by RA) attribute of the response, and puts the source IP address and UDP port of the request (as seen by the TURN server) into the XOR-MAPPED-ADDRESS attribute (denoted by MA). In step 7, this message is sent back to the TURN client and relayed by the NAT in step 8.

The client now proceeds to perform a basic SIP call setup. In message 9, the TURN client includes the TURN server's address (which it learned in message 8) in the SDP of its INVITE (e.g., using syntax described in[I-D.ietf-mmusic-ice] (Rosenberg, J., “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” October 2007.)). The called party responds with its SDP in a provisional response (18x) or a final response (200 Ok). The called party's SDP includes its IP address and UDP port, 192.0.2.17:12734. Immediately after sending its 200 Ok, the called party sends an RTP packet to the TURN server's IP address (11). This RTP packet is dropped by the TURN server, because the TURN server has not been given permission to relay that data. Incoming packets are dropped until a permission is created. The SIP exchange completes with an SIP 200 Ok message (12).

Steps 13-20 show the client performing a channel allocation. The TURN client needs to send an RTP packet. Since no channels and no permissions have been created, the TURN client sends the RTP packet inside of a Send Indication, using channel number 0, with the CHANNEL-NUMBER attribute set to the channel number the TURN client wants to use for subsequent communication with this TURN peer (77 is shown in the example). The TURN peer's IP address and UDP port (which were learned from the SDP answer received in step 10) are placed in the PEER-ADDRESS attribute (denoted by PA). In message 13, the TURN client sends this Send Indication, and it is relayed by the NAT to the TURN server (14).

Upon receipt of that message, the TURN server creates a permission, which allows subsequent traffic from that same peer address to be relayed to that TURN client's IP address and UDP port. The TURN server sends the contents of the Send Indication's DATA attribute towards the PEER-ADDRESS (15); this will typically be an RTP packet. Note that the source address and port of message 15 is the TURN server's address, 192.0.2.3:32766, which is the allocated transport address communicated to the TURN client in messages 7 and 8.
In step 16, the TURN server sends a channel confirmation message to the TURN client. Once the TURN client receives this message, it can forgo using the Send Indication for that channel. Instead, it can utilize the channel number in the TURN framing header. Steps 18 and 19 show the TURN client sending a message to TURN server using the TURN framing header, with channel=1. Step 20 shows the TURN server removing the TURN framing and sending the RTP packet to the TURN peer.
Steps 21-28 show an RTP packet from the TURN peer, which causes a channel allocation by the TURN server. In packet 21, an RTP packet is sent by the TURN peer to the TURN server. There is an existing permission (created in step 14), so the TURN server accepts this incoming RTP packet. The TURN server knows the TURN client to send this packet to, but does not yet have a channel assigned for traffic in this direction. The TURN server chooses a channel number (33 in the example), and sends a Data Indication to the TURN client (message 22). The NAT relays this to the TURN client (message 23). The TURN client sends an Channel Confirmation message (24) which is relayed by the NAT (25). When the TURN server receives the Channel Confirmation, it no longer needs to use a Send Indication for traffic from that remote peer; instead, it can use TURN framing with its chosen channel number (33). The next RTP packet that arrives from that peer (26) is sent by the TURN server using TURN framing indicating the channel number (message 27) and relayed by the NAT to the TURN client (28).

---

15. Changes since version -04

This section lists the major changes between thiis document and draft-ietf-behave-turn-04:

- *Removed the ability to allocate addresses for TCP relaying. This is now covered in a separate document. However, communication between the client and the server can still run over TCP or TLS/TCP. This resulted in the removal of the Connect method and the TIMER-VAL and CONNECT-STAT attributes.

- *Added the concept of channels. All communication between the client and the server flows on a channel. Channels are numbered 0..65535. Channel 0 is used for TURN messages, while the remaining channels are used for sending unencapsulated data to/ from a remote peer. This concept adds a new Channel Confirmation method and a new CHANNEL-NUMBER attribute. The new attribute is also used in the Send and Data methods.

*The framing mechanism formally used just for stream-oriented transports is now also used for UDP, and the former Type and Reserved fields in the header have been replaced by a Channel Number field. The length field is zero when running over UDP.

*TURN now runs on its own port, rather than using the STUN port. The use of channels requires this.

*Removed the SetActiveDestination concept. This has been replaced by the concept of channels.

*Changed the allocation refresh mechanism. The new mechanism uses a new Refresh method, rather than repeating the Allocation transaction.

*Changed the syntax of SRV requests for secure transport. The new syntax is "_turns._tcp" rather than the old "_turn._tls". This change mirrors the corresponding change in STUN SRV syntax.

*Renamed the old REMOTE-ADDRESS attribute to PEER-ADDRESS, and changed it to use the XOR-MAPPED-ADDRESS format.

*Changed the RELAY-ADDRESS attribute to use the XOR-MAPPED-ADDRESS format (instead of the MAPPED-ADDRESS format)).

*Renamed the 437 error code from "No Binding" to "Allocation Mismatch".

*Added a discussion of what happens if a client's public binding on its outermost NAT changes.

*The document now consistently uses the term "peer" as the name of a remote endpoint with which the client wishes to communicate.

*Rewrote much of the document to describe the new concepts. At the same time, tried to make the presentation clearer and less repetitive.

16. Acknowledgements

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

17. References

TOC

17.1. Normative References

TOC

[I-D.ietf-behave-rfc3489bis]	Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, “Session Traversal Utilities for (NAT) (STUN),” draft-ietf-behave-rfc3489bis-18 (work in progress), July 2008 (TXT).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

17.2. Informative References

TOC

[RFC3550]	Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” STD 64, RFC 3550, July 2003 (TXT, PS, PDF).
[RFC3264]	Rosenberg, J. and H. Schulzrinne, “An Offer/Answer Model with Session Description Protocol (SDP),” RFC 3264, June 2002 (TXT).
[RFC4302]	Kent, S., “IP Authentication Header,” RFC 4302, December 2005 (TXT).
[RFC4303]	Kent, S., “IP Encapsulating Security Payload (ESP),” RFC 4303, December 2005 (TXT).
[RFC3424]	Daigle, L. and IAB, “IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation,” RFC 3424, November 2002 (TXT).
[I-D.ietf-mmusic-ice]	Rosenberg, J., “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” draft-ietf-mmusic-ice-19 (work in progress), October 2007 (TXT).
[RFC4787]	Audet, F. and C. Jennings, “Network Address Translation (NAT) Behavioral Requirements for Unicast UDP,” BCP 127, RFC 4787, January 2007 (TXT).

[I-D.ietf-behave-turn-tcp]	Perreault, S. and J. Rosenberg, “Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations,” draft-ietf-behave-turn-tcp-06 (work in progress), March 2010 (TXT).

Authors' Addresses

TOC

	Jonathan Rosenberg
	Cisco Systems, Inc.
	Edison, NJ
	US
Email:	jdrosen@cisco.com
URI:	http://www.jdrosen.net
	Rohan Mahy
	Plantronics, Inc.
Email:	rohan@ekabal.com
	Philip Matthews
	Avaya, Inc.
	1135 Innovation Drive
	Ottawa, Ontario K2K 3G7
	Canada
Phone:	+1 613 592-4343 x223
Fax:	
Email:	philip_matthews@magma.ca
URI:	
	Dan Wing
	Cisco Systems, Inc.
	170 West Tasman Drive
	San Jose, CA 95134
	USA
Phone:	
Fax:	
Email:	dwing@cisco.com
URI:	

Full Copyright Statement

TOC

Intellectual Property