MIDCOM
Internet-Draft
Expires: March 13, 2006

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

September 9, 2005

Traversal Using Relay NAT (TURN)
draft-rosenberg-midcom-turn-08

Status of this Memo

Copyright Notice

Abstract

Traversal Using Relay NAT (TURN) is a protocol that allows for an element behind a NAT or firewall to receive incoming data over TCP or UDP connections. It is most useful for elements behind symmetric NATs or firewalls that wish to be on the receiving end of a connection to a single peer. TURN does not allow for users to run

servers on well known ports if they are behind a nat; it supports the connection of a user behind a nat to only a single peer. In that regard, its role is to provide the same security functions provided by symmetric NATs and firewalls, but to "turn" them into port-restricted NATs.

Table of Contents

1. Introduction

Network Address Translators (NATs), while providing many benefits, also come with many drawbacks. The most troublesome of those drawbacks is the fact that they break many existing IP applications, and make it difficult to deploy new ones. Guidelines [9] have been developed that describe how to build "NAT friendly" protocols, but many protocols simply cannot be constructed according to those guidelines. Examples of such protocols include multimedia applications and file sharing.

Simple Traversal of UDP Through NAT (STUN) [1] provides one means for an application to traverse a NAT. STUN allows a client to obtain a transport address (and IP address and port) which may be useful for receiving packets from a peer. However, addresses obtained by STUN may not be usable by all peers. Those addresses work depending on the topological conditions of the network. Therefore, STUN by itself cannot provide a complete solution for NAT traversal.

A complete solution requires a means by which a client can obtain a transport address from which it can receive media from any peer which can send packets to the public Internet. This can only be accomplished by relaying data though a server that resides on the public Internet. This specification describes Traversal Using Relay NAT (TURN), a protocol that allows a client to obtain IP addresses and ports from such a relay.

Although TURN will almost always provide connectivity to a client, it comes at high cost to the provider of the TURN server. It is therefore desirable to use TURN as a last resort only, preferring other mechanisms (such as STUN or direct connectivity) when possible. To accomplish that, the Interactive Connectivity Establishment (ICE) [13] methodology can be used to discover the optimal means of connectivity.

2. Terminology

In this document, the key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL are to be interpreted as described in RFC 2119 [2] and indicate requirement levels for compliant TURN implementations.

3. Definitions

TURN Client: A TURN client (also just referred to as a client) is an entity that generates TURN requests. A TURN client can be an end system, such as a Session Initiation Protocol (SIP) [6] User Agent, or can be a network element, such as a Back-to-Back User

Agent (B2BUA) SIP server. The TURN protocol will provide the TURN client with IP addresses that route to it from the public Internet.

TURN Server: A TURN Server (also just referred to as a server) is an entity that receives TURN requests, and sends TURN responses. The server is capable of acting as a data relay, receiving data on the address it provides to clients, and forwarding them to the clients.

Transport Address: An IP address and port.

4. Applicability Statement

TURN is useful for applications that require a client to place a transport address into a protocol message, with the expectation that the client will be able to receive packets from a single host that will send to this address. Examples of such protocols include SIP, which makes use of the Session Description Protocol (SDP) [7]. SDP carries and IP address on which the client will receive media packets from its peer. Another example of a protocol meeting this criteria is the Real Time Streaming Protocol (RTSP) [8].

When a client is behind a NAT, transport addresses obtained from the local operating system will not be publically routable, and therefore, not useful in these protocols. TURN allows a client to obtain a transport address, from a server on the public Internet, which can be used in protocols meeting the above criteria. However, the transport addresses obtained from TURN servers are not generally useful for receiving data from anywhere. They are only useful for communicating with a single peer. This is accomplished by having the TURN server emulate the behavior of an address-restricted NAT. In particular, the TURN server will only relay packets from an external IP address towards the client if the client had previously sent a packet through the TURN server towards that IP address. As a result of this, when a TURN server is placed in front of a symmetric NAT, the resulting combined system has identical security properties to a system that just had an address restricted NAT. Since clients behind such devices cannot run public servers, they cannot run them behind TURN servers either.

5. Overview of Operation

The typical TURN configuration is shown in Figure 1. A TURN client is connected to private network 1. This network connects to private network 2 through NAT 1. Private network 2 connects to the public Internet through NAT 2. On the public Internet is a TURN server.

```
                              /-----\
                            // TURN  \\
                           |   Server  |
                            \\       //
                              \-----/


                         +--------------+            Public Internet
         ................|     NAT 2    |.......................
                         +--------------+

                         +--------------+            Private NET 2
         ................|     NAT 1    |.......................
                         +--------------+

                             /-----\
                           //  TURN \\
                          |    Client |
                           \\       //               Private NET 1
                             \-----/


                                 Figure 1
```

TURN is a simple client-server protocol. It is identical in syntax and general operation to STUN, in order to facilitate a joint implementation of both. TURN defines a request message, called Allocate, which asks the TURN server to allocate a public IP address and port. TURN can run over UDP and TCP, as it allows for a client to request address/port pairs for receiving both UDP and TCP.

A TURN client first discovers the address of a TURN server. This can be preconfigured, or it can be discovered using SRV records [3] This will allow for different TURN servers for UDP and TCP. Once a TURN server is discovered, the client sends a TURN Allocate request to the TURN server. TURN provides a mechanism for mutual authentication and integrity checks for both requests and responses, based on a shared secret. Assuming the request is authenticated and has not been tampered with, the TURN server allocates a transport address to the TURN client, called the allocated transport address, and returns it in the response to the Allocate Request. Normally, the allocated transport address will be on one of the interfaces on the TURN server itself. However, it is also allowed for the TURN server to be behind a NAT, in which case the allocated transport address may correspond to the NAT, which is then mapped to the private address of the TURN server. Proper operation of the TURN server will require it to have many bindings established in the NAT ahead of time; the means for doing so are outside the scope of this specification.

However, the TURN server will not relay any packets from PA to SA until the client sends a packet through the TURN server towards a correspondent. To do that, a client sends a TURN Send command, which includes a data packet and a destination IP address and port. The TURN server, upon receipt of this command, will forward the packet to that IP address and port, add a "permission" for that IP address, so that inbound packets from that address and port are permitted. In the case of TCP, the Send Request will cause the TURN server to open a TCP connection towards the target if one is not already up.

Packets received from the TURN server via UDP or via the TCP connections opened by a Send Request are then forwarded towards the client, encapsulated in Data Indication messages. The usage of Send Requests and Data Indication messages is inefficient. As a result, once the client has concluded on a specific external client with which it wishes to communicate, it can issue a Set Active Destination Request. This request informs the server of a destination for which unencapsulated packets are to be forwarded. As such, if a client sends a packet to the TURN server which is not a TURN packet, it gets sent to this destination. Similarly, packets from the external client to the TURN server are forwarded to the client without encapsulation in a Data Indication message.

Once an active destination is set, it cannot be changed for TCP. Effectively, the TCP connection from the client to the TURN server "switches" to the application's ownership once the Set Active Destination Request has been received.

To do all of this, the TURN server will maintain a binding between an internal 5-tuple and 1 or more external 5-tuples, as shown in Figure 2. The internal 5-tuple represents the "connection" between the TURN server and the TURN client. It is the actual connection in the case of TCP, and in the case of UDP, it is the combination of the IP address and port from which the TURN client sent its Allocate Request, with the IP address and port to which that Allocate Request 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 TURN client is communicating with through the TURN server. Initially, there aren't any external 5-tuples, since the TURN 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.

```
                                                               +---------+
                                                               |         |
                                                               | External|
                                                             / | Client  |
                                                           //  |         |
                                                          /    |         |
                                                        //     +---------+
                                                       /
                                                     //
                     +-+                            /
                     | |                           /
                     | |                         //
      +---------+    | |          +---------+   /              +---------+
      |         |    |N|          |         | //               |         |
      | TURN    |    | |          |         |/                 | External|
      | Client  |----|A|----------|   TURN  |------------------| Client  |
      |         |    | |^        ^|  Server |^                ^|         |
      |         |    |T||        ||         ||                ||         |
      +---------+    | ||        |+---------+|                |+---------+
         ^           | ||        |           |                |
         |           | ||        |           |                |
         |           +-+|        |           |                |
         |              |        |           |                |
         |
                    Internal     Internal    External         External
     Client         Remote       Local       Local            Remote
     Performing     Transport    Transport   Transport        Transport
     Allocations    Address      Address     Address          Address

                        |          |            |                |
                        +-----+----+            +--------+-------+
                              |                          |
                              |                          |

                            Internal                External
                            5-Tuple                 5-tuple
```

Figure 2

For TCP, the TURN server does not need to examine the data received; it merely forwards all data between the socket pairs it has associated together. In the case of UDP, the TURN server looks for a magic cookie in the first 128 bytes of each UDP packet. If present, it indicates that the packet is a TURN control packet, used for keepalives and teardown of the binding. In the case of TCP, if

either side closes a connection, the TURN server closes the other connection. For both UDP and TCP, the TURN server can also time out a connection in the event data is not received after some configured time out period. This period is sent to the client in the TURN response to the Allocate request.

A TURN server will accept UDP packets and TCP connections from external clients only when a permission for accepting it has been created at the TURN server. The client creates this permission by sending a packet to a specific transport address through the TURN server.

6. Message Overview

TURN messages are identical to STUN messages in their syntax. TURN defines several new messages - the Allocate Request, the Allocate Response, the Allocate Error Response, the Send Request, the Send Response, the Send Error Response, the Set Active Destination Request, Set Active Destination Response, Set Actice Destination Error Response and the Data Indication. TURN also uses the Shared Secret Request, Shared Secret Response, and Shared Secret Error Response defined by STUN. TURN makes use of some of the STUN attributes (MAPPED-ADDRESS, USERNAME, MESSAGE-INTEGRITY, ERROR-CODE, and UNKNOWN-ATTRIBUTES) and also defines several of its own. Specifically, TURN adds the LIFETIME attribute, which allows the TURN server to tell the client when the binding will be released. It defines the ALTERNATE-SERVER attribute, which allows the server to redirect the TURN client to connect to an alternate server. It defines the MAGIC-COOKIE attribute, which allows the TURN client to find TURN messages in a stream of UDP packets. It defines the BANDWIDTH attribute, which allows a client to inform the server of the expected bandwidth usage on the connection. It defines the DESTINATION-ADDRESS attribute, which is used in the Send Request to identify where the data should be sent to. It defines the REMOTE-ADDRESS, which appears in a Data Indication, and tells the client where the data came from. It defines the DATA attribute, which contains the content in a Data Indication. Finally, it defines the NONCE and REALM attributes, used for authentication.

7. Server Behavior

The server behavior depends on whether the request is a Shared Secret Request, an Allocate Request or a Set Active Destination Request.

7.1 Shared Secret Request

Unlike a STUN server, a TURN server provides resources to clients that connect to it. Therefore, only authorized clients can gain

access to a TURN server. This requires that TURN requests be authenticated. TURN assumes the existence of a long-lived shared secret between the client and the TURN server in order to achieve this authentication. The client uses this long-lived shared secret to authenticate itself in a Shared Secret Request, sent over TLS. The Shared Secret Response provides the client with a one-time username and password. This one-time credential is then used by the server to authenticate an Allocate Request. The usage of a separate long lived and one-time credentials prevents dictionary attacks, whereby an observer of a message and its HMAC could guess the password by an offline dictionary search.

When a TURN server receives a Shared Secret Request, it first executes the processing described in the first three paragraphs of Section 8.2 of STUN. This processing will ensure that the Shared Secret Request is received over TLS.

Assuming it was, the server checks the Shared Secret Request for a MESSAGE-INTEGRITY attribute. If not present, the server generates a Shared Secret Error Response with an ERROR-CODE attribute with response code 401. That response MUST include a NONCE attribute, containing a nonce that the server wishes the client to reflect back in a subsequent Shared Secret Request (and therefore include the message integrity computation). The response MUST include a REALM attribute, containing a realm from which the username and password are scoped [4].

If the MESSAGE-INTEGRITY attribute was present, the server checks for the existence of the REALM attribute. If the attribute is not present, the server MUST generate a Shared Secret Error Response. That response MUST include an ERROR-CODE attribute with response code 434. That response MUST include a NONCE and a REALM attribute.

If the REALM attribute was present, the server checks for the existence of the NONCE attribute. If the NONCE attribute is not present, the server MUST generate a Shared Secret Error Response. That response MUST include an ERROR-CODE attribute with response code 435. That response MUST include a NONCE attribute and a REALM attribute.

If the NONCE attribute was present, the server checks for the existence of the USERNAME attribute. If it was not present, the server MUST generate a Shared Secret Error Response. The Shared Secret Error Response MUST include an ERROR-CODE attribute with response code 432. It MUST include a NONCE attribute and a REALM attribute.

If the USERNAME is present, the server computes the HMAC over the

request as described in Section 11.2.8 of STUN. The key is computed as MD5(unq(USERNAME-value) ":" unq(REALM-value) ":" passwd), where the password is the password associated with the username and realm provided in the request. If the server does not have a record for that username within that realm, the server generates a Shared Secret Error Response. That response MUST include an ERROR-CODE attribute with response code 436. That response MUST include a NONCE attribute and a REALM attribute.

This format for the key was chosen so as to enable a common authentication database for SIP and for TURN, as it is expected that credentials are usually stored in their hashed forms.

If the computed HMAC differs from the one from the MESSAGE-INTEGRITY attribute in the request, the server MUST generate a Shared Secret Error Response with an ERROR-CODE attribute with response code 431. This response MUST include a NONCE attribute and a REALM attribute.

If the computed HMAC doesn't differ from the one in the request, but the nonce is stale, the server MUST generate a Shared Secret Error Response. That response MUST include an ERROR-CODE attribute with response code 430. That response MUST include a NONCE attribute and a REALM attribute.

In all cases, the Shared Secret Error Response is sent over the TLS connection on which the Shared Secret Request was received.

The server proceeds to authorize the client. The means for authorization are outside the scope of this specification. It is anticipated that TURN servers will be run by providers that also provide an application service, such as SIP or RTSP. In that case, a user would be authorized to use TURN if they are authorized to use the application service.

The server then generates a Shared Secret Response as in Section 8.2 of STUN. This response will contain a USERNAME and PASSWORD, which are used by the client as a short-term shared secret in subsequent Allocate requests. Note that STUN specifies that the server has to invalidate this username and password after 30 minutes. This is not the case in TURN. In TURN, the server MUST store the allocated username and password for a duration of at least 30 minutes. Once an Allocate request has been authenticated using that username and password, if the result was an Allocate Error Response, the username and password are discarded. If the result was an Allocate Response, resulting in the creation of a new binding, the username and password become associated with that binding. They can only be used to authenticate Allocate requests sent from the same source transport address in order to refresh or de-allocate that binding. Once the

binding is deleted, the username and password are discarded.

This policy avoids replay attacks, whereby a recorded Allocate request is replayed in order to obtain a binding without proper authentication. It also ensures that existing bindings can be refreshed without needed to continuously obtain one-time passwords from the TURN server.

7.2 Allocate Request

7.2.1 Overview

Allocate requests are used to obtain an IP address and port that the client can use to receive UDP and TCP packets from any host on the network, even when the client is behind a symmetric NAT. To do this, a TURN server allocates a local transport address, and passes it to the client in an Allocate Response. The TURN server has a configured policy that defines whether or not a packet received from an external client will be passed to the TURN client. This is a set of IP addresses and optionally ports that identify the permitted external clients. This set of addresses is built up as a consequence of Send requests from the TURN client.

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 matches the internal remote and local transport addresses of an existing internal 5-tuple. A subsequent request is one whose source and destination transport address do not match the internal remote and local transport address of an existing internal 5-tuple.

7.2.2 Initial Requests

A TURN server MUST be prepared to receive Allocate Requests over TCP and UDP. The port on which to listen is based on the DNS SRV entries provided by the server. Typically, this will be XXXX, the default TURN port.

The server MUST check the Allocate Request for a MESSAGE-INTEGRITY attribute. If not present, the server generates a Allocate Error Response with an ERROR-CODE attribute with response code 401.

If the MESSAGE-INTEGRITY attribute was present, the server checks for the existence of the USERNAME attribute. If it was not present, the server MUST generate a Allocate Error Response. The Allocate Error Response MUST include an ERROR-CODE attribute with response code 432.

If the USERNAME is present, the server computes the HMAC over the

request as described in Section 11.2.8 of STUN. The key is equal to the password associated with the username in the request, where that username is a short term username allocated by the TURN server. The username MUST be one which has been allocated by the server in a Shared Secret Response, but has not yet been used to authenticate an Allocate request. If that username is not known by the server, or has already been used, the server generates an Allocate Error Response. That response MUST include an ERROR-CODE attribute with response code 430.

If the computed HMAC differs from the one from the MESSAGE-INTEGRITY attribute in the request, the server MUST generate a Allocate Error Response with an ERROR-CODE attribute with response code 431.

Assuming the message integrity check passed, processing continues. The server MUST check for any attributes in the request with values less than or equal to 0x7fff which it does not understand. If it encounters any, the server MUST generate an Allocate Error Response, and it MUST include an ERROR-CODE attribute with a 420 response code.

That response MUST contain an UNKNOWN-ATTRIBUTES attribute listing the attributes with values less than or equal to 0x7fff which were not understood.

If the Allocate request arrived over TCP, the Allocate Error Response is sent on the connection from which the request arrived. If the Allocate request arrived over UDP, the Allocate Error Response is sent to the transport address from which the request was received (i.e., the source IP address and port), and sent from the transport address on which the request was received (i.e., the destination IP address and port).

Assuming the Allocate request was authenticated and was well-formed, 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 port for the client. If the clients source port was in the range 1024-65535, it is RECOMMENDED that the server allocate a port in that range. If the clients source port was in the range of 1-1024, port selection is at the discrtion of the administrator. It is RECOMMENDED that a port in the range of 1024-65535 be allocated. This is one of several ways to prohibit TURN from being used to attempt to run standard services. These guidelines are meant to be consistent with [14], since the TURN relay is effectively a NAT.

If a port meeting the bandwidth constraints cannot be allocated, the

server MUST generate a Allocate Error Response that includes an ERROR-CODE attribute with a response code of 300. That response MAY include an ALTERNATE-SERVER attribute pointing to an alternate server which can be used by the client.

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.

The server MUST remember the one-time username and password used to obtain the allocated transport address, and also associate it with the internal 5-tuple.

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 binding, 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 is set to fire at or after that time. Section 7.7 discusses behavior when the timer fires.

Once the port has been obtained from the operating system and the activity timer started for the port binding, the server generates an Allocate Response. The Allocate Response MUST contain the same transaction ID contained in the Allocate Request. The length in the message header MUST contain the total length of the message in bytes, excluding the header. The Allocate Response MUST have a message type of "Allocate Response".

The response MUST contain a MAGIC-COOKIE as the first attribute (this is done so that endpoints can consistently use the presence of MAGIC-COOKIE to discern TURN packets). The server MUST add a MAPPED-ADDRESS attribute to the Allocate Response, and set it to the allocated transport address.

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

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 10.

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.

The TURN server then sends the response. If the Allocate request was received over TCP, the response is sent over that TCP connection. If the Allocate request was received over UDP, the response is sent to the transport address from which the request was received (i.e., the source IP address and port), and sent from the transport address on which the request was received (i.e., the destination IP address and port).

If the allocated port for TCP, the server MUST be prepared to receive a TCP connection request on that port.

7.2.3 Subsequent Requests

Once a binding has been created for UDP and permissions installed, the client can send subsequent Allocate requests to the TURN server. To determine which packets are for the TURN server, and which need to be relayed, the server looks at the packet. If the packet is shorter than 28 bytes, it is not a TURN request. If it is longer than 28 bytes, the server checks bytes 25-28. If these bytes are equal to the MAGIC-COOKIE, the request is a TURN request. Otherwise, it is a data packet, and is to be relayed.

The server first authenticates the request. This is done as in Section 7.2.2. The request MUST be authenticated using the same one-time username and password used previously. That is, the source and destination transport address of the Allocate Request are compared, respectively, with the internal remote and local transport addresses associated with existing allocations. If there is a match, the same username and password used to obtain that allocation must match the ones used in the request. If there is not a match, the server MUST generate an Allocate Error Response with a 441 response code.

The server looks for the LIFETIME attribute in the Allocate Request. If not found, it determines the default refresh duration, in seconds, for this binding. If the LIFETIME attribute was present in the request, and the value is larger than the maximum duration the server is willing to extend the lifetime of the binding, the server MAY lower it to that maximum. However, the server MUST NOT increase the duration requested in the LIFETIME attribute. The resulting duration is added to the current time, and the activity timer for this binding

is reset to fire at or after that time. Section 7.7 discusses behavior when the timer fires.

Once the timer is set, the server MUST generate an Allocate Response. The Allocate Response MUST contain the same transaction ID contained in the Allocate Request. The length in the message header MUST contain the total length of the message in bytes, excluding the header. The Allocate Response MUST have a message type of "Allocate Response". The response MUST contain a MAGIC-COOKIE as the first attribute. It MUST contain a MAPPED-ADDRESS which contains the allocated transport address. It MUST contain a LIFETIME attribute which contains the time from now until the point at which the binding will be deleted. The final attribute MUST be a MESSAGE-INTEGRITY attribute, which MUST use the same one-time username and password used to authenticate the request.

The TURN server then sends the response. The response is sent to the transport address from which the request was received (i.e., the source IP address and port), and sent from the transport address on which the request was received (i.e., the destination IP address and port).

Subsequent allocation requests cannot be sent for TCP, and as such, the server should never receive them. Indeed, a server MUST NOT look for them in the TCP data stream.

7.3 Send Request

A Send request 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. Send requests are used with both UDP and TCP. A server can disambiguate a UDP Send Request from a data packet by looking for the MAGIC-COOKIE attribute, as described in Section 7.2.3. Such disambiguation is not needed for TCP, since the client cannot send this request after an external 5-tuple has been activated.

Once the server has identified a request as a Send request, the server verifies that it has arrived with a source and destination transport address that matches the internal remote and local transport address of an internal 5-tuple associated with an existing allocation. If there is no matching allocation, the server MUST generate a 437 (No Binding) Send Error Response.

Next, the server authenticates the request. This is done as in Section 7.2.2. The request MUST be authenticated using the same one-time username and password used previously. That is, the source and destination transport address of the Allocate Request are compared,

respectively, with the internal remote and local transport addresses associated with existing allocations. If there is a match, the same username and password used to obtain that allocation must match the ones used in the request. If there is not a match, the server MUST generate a Send Error Response with a 441 response code.

Once the request has been authenticated, the server validates it. The request should contain a DESTINATION-ADDRESS attribute and a DATA attribute. If it doesn't, the server MUST reject the request with a 400 (Bad Request) Send Error Response.

Assuming the Send Request has been validated, the server then takes the contents of the DATA attribute. In the case of UDP, it creates a UDP packet whose payload equals that content. The server sets the source IP address equal to the allocated transport address. The destination transport address is set to the contents of the DESTINATION-ADDRESS attribute. 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 clients responsibility to generate another Send Request if needed. If the TURN client hasn't previously sent to this destination IP address and port, an external 5-tuple is instantiated in the TURN server. Its local and remote transport addresses, respectively, are set to the source and destination transport addresses of the UDP packet.

In the case of TCP, the server checks if it has an existing TCP connection open from the allocated transport address to the address in the DESTINATION-ADDRESS attribute. If so, the server extracts the content of the DATA attribute and sends it on matching TCP connection. If the server doesn't have an existing TCP connection to the destination, it MUST open one from an epheral port on the same interface as the allocated transport address, and do so to the transport address in the DESTINATION-ADDRESS attribute. Once the connection is established, the server MUST send the contents of the DATA attribute onto that connection. If the connection could not be opened, or if the transmission of the data resulted in an error, the TURN server MUST generate a Send Error Response with a 438 (Send Failed) response code.

If the UDP packet or TCP data was sent without errors, the server generates a Send Response. The Send Response MUST have a message type of "Send Response". The response MUST contain a MAGIC-COOKIE as the first attribute and a MESSAGE-INTEGRITY attribute as the last. If the server needs to generate a Send Error Response, that message MUST contain a message type of "Send Error Response", and MUST contain a MAGIC-COOKIE as the first attribute. It MUST contain an ERROR-CODE with the appropriate response code. For UDP, both the Send Response and Send Error Response are sent back to the source IP

and port where the request came from, and sent from the same address and port where the request was sent to. For TCP, the response is sent on the TCP connection that the server received the request on.

The server then adds the IP address of the DESTINATION-ADDRESS attribute to the permission list for this allocation. This happens regardless of whether, in the case of TCP, the data was sent successfully.

7.4 Receiving Packets and Connections on the Allocated Transport Address

If a TURN server receives a TCP connection request on an allocated transport address, it checks the permissions associated with that allocation. If the source IP address of the TCP SYN packet match one of the permissions, the TCP connection is accepted. Otherwise, it is rejected.

If a TURN server receives a UDP packet on an allocated 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 UDP packet is accepted. Otherwise, it is discarded.

> This emulates the address-restricted behavior of a NAT, as opposed to the stricter port and address restricted behavior. This allows for interoperation of TURN with clients that don't perform symmetric RTP, and is needed to avoid double relays for sessions between clients that are both behind symmetric NAT.

If the source and destination transport address of the UDP packet is equal, respectively, to the remote and local transport addresses of the active 5-tuple, the UDP packet is forwarded to the client and not encapsulated in a TURN packet. To forward, the packet is sent with a source IP address and port equal to the internal local transport address, and with a destination address and port equal to the internal remote transport address.

Similarly, if data is received on a TCP connection, and that connection is the active connection, the data on that connection is copied onto the connection to the TURN client.

If data is received from an external client on a TCP connection, and that connection is not the active connection, the data are sent to the client in a Data Indication message. The Data Indication message MUST contain a MAGIC-COOKIE attribute as the first attribute. The Data Indication message MUST contain a DATA attribute whose contents are equal to the data just received on the TCP connection from the external client. The message MUST contain a REMOTE-ADDRESS attribute

whose content is equal to the external remote transport address. This packet is sent to the TURN client over the TCP connection to the TURN client.

If a UDP packet is received from an external client, and the external 5-tuple don't match the active 5-tuple, the data is sent to the client in a Data Indication message. This message is not retransmitted by the server, and which does not generate a response. As a result, like data packets which are forwarded, there is no reliability guarantee provided by the TURN server for this indication. The Data Indication message MUST contain a MAGIC-COOKIE attribute as the first attribute. It MUST contain a DATA attribute whose contents are equal to the payload of the UDP packet. The message MUST contain a REMOTE-ADDRESS attribute whose content is equal to the source IP address and port of the UDP packet received by the TURN server. This packet is sent to the internal remote transport address, and sent from the internal local transport address.

Note that, because data is forwarded blindly across TCP bindings, TLS will successfully operate over a TURN allocated TCP port.

7.5 Receiving a Set Active Destination Request

The Set Active Destination Request is used by a client to determine an external 5-tuple that will be used as the forwarding destination of all non-TURN data from the TURN client. Furthermore, all data from that external client will be forwarded to the TURN client without encapsulation in a Data Indication.

A server can disambiguate a UDP Set Active Destination Request from a data packet by looking for the MAGIC-COOKIE attribute, as described in Section 7.2.3. Such disambiguation is not needed for TCP, since the client cannot send this request more than once.

The active destination is initially null. It is always explicitly set by the Set Active Destination Request.

Once the server has identified a request as a Set Active Destination request, the server verifies that it has arrived with a source and destination transport address that matches the internal remote and local transport address of an internal 5-tuple associated with an existing allocation. If there is no matching allocation, the server MUST generate a 437 (No Binding) Send Error Response.

Next, the server authenticates the request. This is done as in Section 7.2.2. The request MUST be authenticated using the same one-time username and password used previously. That is, the source and

destination transport address of the Allocate Request are compared, respectively, with the internal remote and local transport addresses associated with existing allocations. If there is a match, the same username and password used to obtain that allocation must match the ones used in the request. If there is not a match, the server MUST generate a Set Active Destination Error Response with a 441 response code.

Once the request has been authenticated, the server validates it. The request should contain a DESTINATION-ADDRESS attribute. If it doesn't, the server MUST reject the request with a 400 (Bad Request) Set Active Destination Error Response.

If there is already an active 5-tuple, and the external remote transport address of that 5-tuple matches the DESTINATION-ADDRESS, the request is basically a no-op. The server MUST generate a Set Active Destination Response. This response contains no attributes. If there is an active 5-tuple but its external remote transport address doesn't match, the request is asking for a change in the destination. The server checks its existing external 5-tuples for one whose external remote transport address matches the DESTINATION-ADDRESS. If none is found, the request is rejected with a 440 (No Destination) response. If one is found, the currently active one is deactivated. A timer, Ta, is set to fire in 3 seconds. The server sets its state to "transitioning". When Ta fires, the server returns to normal operations, and then sets the external 5-tuple that matched the DESTINATION-ADDRESS to the active one. While in the transitioning state, the server behaves as described in this specification, except for the processing of the Set Active Destination Request as described below.

If there is no active 5-tuple, but the server is in the transitioning state, it MUST reject the request with a Set Active Destination Error Response that includes a 439 (Transitioning) response code.

If there is no active 5-tuple, and the server is not in the transitioning state, the server checks its existing external 5-tuples for one whose external remote transport address matches the DESTINATION-ADDRESS. If one is found, that external 5-tuple is made the active one, and a Set Active Destination Response is sent. If none is found, the request is rejected with a 440 (No Destination) response.

If the server needs to send a Set Active Destination Response, that message MUST contain a message type of "Set Active Destination Response", and MUST contain a MAGIC-COOKIE as the first attribute and a MESSAGE-INTEGRITY as the last. If the server needs to send a Set Active Destination Error Response, that message MUST contain a

message type of "Set Active Destination Error Response", and MUST contain a MAGIC-COOKIE as the first attribute. It MUST contain an ERROR-CODE with the appropriate response code. For UDP, both the responses are sent back to the source IP and port where the request came from, and sent from the same address and port where the request was sent to. For TCP, the response is sent on the TCP connection that the server received the request on.

7.6 Receiving Data from the TURN Client

If a TURN server receives a UDP packet from the TURN client on its internal local transport address, and it is coming from an internal remote transport address associated with an existing allocation, and the UDP packet is not a TURN packet (known by the absence of the MAGIC-COOKIE attribute), it represents UDP data that the client wishes to forward. If there is an active 5-tuple, the TURN server MUST forward the UDP packet on that 5-tuple. The destination address and port of the UDP packet is set to the external remote transport address of the active 5-tuple. The source IP address and port of the UDP packet is set to the external local transport address of the active 5-tuple. The TURN server SHOULD NOT retransmit the packet once it has forwarded it. Such retransmissions are the responsibility of the client.

If there is no active 5-tuple, the UDP packet is discarded.

If data is received on a TCP connection from the TURN client, and the previous TURN message from the client was a Set Active Destination Request, the data is forwarded to the active connection. Note that, once a connection has been activated with a Set Active Destination TURN Request, TURN messaging from the client over its TCP connection to the server is not allowed. Thus, all subsequent data will be non-TURN data by definition.

If a TCP connection associated with an allocated transport address is closed, any connections to external clients MUST be closed. At that point, the binding is destroyed. Similarly, if a connection from the TURN server to an external client is closed, and that connection was the active connection, the corresponding connection to the TURN client MUST be closed, and the binding is destroyed.

7.7 Lifetime Expiration

When the activity timer for a binding fires, the server checks to see if there has been any activity on the binding since its creation, or since the last firing of the timer, whichever is more recent. Activity is defined as connection establishment, or packet transmission in either direction. If there has been activity, the

timer is set to fire once again in M seconds, where M is the value of the LIFETIME attribute returned in the most recent Allocate Response for this binding. Note that, with TCP connections, a client cannot send TURN requests once a connection has been set to active. As such, the value of M that is used will be the one from the first Allocate Request; refreshes will not take place with TURN messages.

If there has been no activity, the server MUST destroy the binding, along with its associated one-time password. If the binding was over TCP, the server MUST close any connections it is holding to the client and to the remote client.

8. Client Behavior

Client behavior is broken into several separate steps. First, the client obtains a one-time username and password. Secondly, it generates initial Allocate Requests, and processes the responses. It manages those addresses (refreshing and tearing them down), issues Send Requests and Set Active Destination Requests, and processes TURN indications and data received on those addresses.

8.1 Discovery

Generally, the client will be configured with a domain name of the provider of the TURN servers. This domain name is resolved to an IP address and port of using the SRV procedures [3]. When sending a Shared Secret request, the service name is "turn" and the protocol is "tcp". RFC 2782 spells out the details of how a set of SRV records are sorted and then tried. However, it only states that the client should "try to connect to the (protocol, address, service)" without giving any details on what happens in the event of failure. Those details are described here for TURN.

For TURN requests, failure occurs if there is a transport failure of some sort (generally, due to fatal ICMP errors in UDP or connection failures in TCP). Failure also occurs if the request does not solicit a response after 9.5 seconds. If a failure occurs, the client SHOULD create a new request, which is identical to the previous, but has a different transaction ID and MESSAGE-INTEGRITY attribute. That request is sent to the next element in the list as specified by RFC~2782.

8.2 Obtaining a One Time Password

In order to allocate addresses, a client must obtain a one-time username and password from the TURN server. A unique username and password are required for each distinct address allocated from the server.

To obtain a one-time username and password, the client generates and sends a Shared Secret Request. This is done as described in Section 9.2 of STUN. This request will have no attributes, and therefore, based on the processing in Section 7.1, the server will reject it with a Shared Secret Error Response with a 401 response code. That response will contain a NONCE and a REALM. The client SHOULD generate a new Shared Secret Request (with a new transaction ID), which contains the NONCE and REALM attributes copied from the 401 response. The request MUST include the USERNAME attribute, which contains a username supplied by the user for the specified realm. The request MUST include a MESSAGE-INTEGRITY attribute as the last attribute. The key for the HMAC is computed as described in Section 7.1.

If the response (either to the initial request or to the second attempt with the credentials) is a Shared Secret Error Response, the processing depends on the value of the response code in the ERROR-CODE attribute. If the response code was a 430, the client SHOULD generate a new Shared Secret Request, using the username and password provided by the user, and the REALM and NONCE provided in the 430 response. For a 431 or 436 response code, the client SHOULD alert the user. For a 432, 434 and 435 response codes, if the client had omitted the USERNAME, REALM or NONCE attributes, respectively, from the previous request, it SHOULD retry, this time including the USERNAME, NONCE, REALM, and MESSAGE-INTEGRITY attributes. For a 500 response code, the client MAY wait several seconds and then retry the request. For a 600 response code, the client MUST NOT retry the request, and SHOULD display the reason phrase to the user. Unknown attributes between 400 and 499 are treated like a 400, unknown attributes between 500 and 599 are treated like a 500, and unknown attributes between 600 and 699 are treated like a 600. Any response between 100 and 399 MUST result in the cessation of request retransmissions, but otherwise is discarded.

If a client receives a Shared Secret Response with an attribute whose type is unknown and greater than 0x7fff, the attribute MUST be ignored. If the client receives a Shared Secret Response with an attribute whose type is unknown and less than or equal to 0x7fff, the response is ignored.

If the response is a Shared Secret Response, it will contain the USERNAME and PASSWORD attributes. The client can use these to authenticate an Allocate Request, as described below.

A client MAY send multiple Shared Secret Requests over the same TLS connection, and MAY do so without waiting for responses to previous requests. The client SHOULD close its connection when it has completed allocating usernames and passwords.

8.3 Allocating a Binding

When a client wishes to obtain a transport address, it sends an Allocate Request to the TURN server. Requests for TCP transport addresses MUST be sent over a TCP connection, and requests for UDP transport addresses MUST be sent over UDP.

First, the client obtains a one-time username and password, using the mechanisms described in Section 8.2. The client then formulates an Allocate Request. The request MUST contain a transaction ID, unique for each request, and uniformly and randomly distributed between 0 and 2**128 - 1. The message type of the request MUST be "Allocate Request". The length is set as described in Section 11.1 of STUN.

The Allocate request MUST contain the MAGIC-COOKIE attribute as the first attribute.

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 binding by including it in the LIFETIME attribute in the request. If the no data is sent or received on the binding before expiration of the lifetime, the binding will be deleted by the client.

The client MUST include a USERNAME attribute, containing a username obtained from a previous Shared Secret Response. The request MUST include a MESSAGE-INTEGRITY attribute as the last attribute. The key is equal to the password obtained from the PASSWORD attribute of the Shared Secret Response. The Allocate Request MUST be sent to the same IP address and port as the Shared Secret Request, but from a different local ephemeral port (in other words, the TCP/TLS connection used to obtain the shared secret is not reused for allocations). This is because one time passwords are expected to be host-specific. Rules for retransmissions for Allocate Requests sent over UDP are identical to those for STUN Binding Requests. Allocate Requests sent over TCP are not retransmitted. Transaction timeouts are identical to those for STUN Binding Requests, independent of the transport protocol.

8.4 Processing Allocate Responses

If the response is an Allocate Error Response, the client checks the response code from the ERROR-CODE attribute of the response. For a 400 response code, the client SHOULD display the reason phrase to the user. For a 420 response code, the client SHOULD retry the request, this time omitting any attributes listed in the UNKNOWN-ATTRIBUTES

attribute of the response. For a 430 response code, the client SHOULD obtain a new one-time username and password, and retry the Allocate Request with a new transaction. For 401 and 432 response codes, if the client had omitted the USERNAME or MESSAGE-INTEGRITY attribute as indicated by the error, it SHOULD try again with those attributes. A new one-time username and password is needed in that case. For a 431 response code, the client SHOULD alert the user, and MAY try the request again after obtaining a new username and password. For a 300 response code, the client SHOULD attempt a new TURN transaction to the server indicated in the ALTERNATE-SERVER attribute. For a 500 response code, the client MAY wait several seconds and then retry the request with a new username and password. For a 600 response code, the client MUST NOT retry the request, and SHOULD display the reason phrase to the user. Unknown attributes between 400 and 499 are treated like a 400, unknown attributes between 500 and 599 are treated like a 500, and unknown attributes between 600 and 699 are treated like a 600. Unknown attributes between 300 and 399 are treated like 300. Any response between 100 and 299 MUST result in the cessation of any request retransmissions, but otherwise is discarded.

If a client receives a response with an attribute whose type is unknown and greater than 0x7fff, the attribute MUST be ignored. If the client receives a response with an attribute whose type is unknown and less than or equal to 0x7fff, any request retransmissions MUST cease, but the entire response is otherwise ignored.

If the response is an Allocate Response, the client MUST check the response for a MESSAGE-INTEGRITY attribute. If not present, the client MUST discard the response. If present, the client computes the HMAC over the response. The key MUST be same as used to compute the MESSAGE-INTEGRITY attribute in the request. If the computed HMAC differs from the one in the response, the client MUST discard the response, and SHOULD alert the user about a possible attack. If the computed HMAC matches the one from the response, processing continues.

The MAPPED-ADDRESS in the Allocate Response can be used by the client for receiving packets. The server will expire the binding after LIFETIME seconds have passed with no activity. The server will allow the user to send and receive no more than the amount of data indicated in the BANDWIDTH attribute.

8.5 Refreshing a Binding

If there has been no activity on a UDP binding for a period of time equalling 3/4 of the lifetime of the binding (as conveyed in the LIFETIME attribute of the Allocate Response), the client SHOULD

refresh the binding with another Allocate Request if it wishes to keep it. Note that only UDP bindings can be refreshed. For TCP, application-specific keepalives are needed.

To perform a refresh, the client generates an Allocate Request as described in Section 8.3. However, the one-time username and password used MUST be the same as those used in the successful Allocate Request for that binding. The client will need to look for the TURN response amongst the data packets using the MAGIC-COOKIE, as described in Section 7.2.3. Processing of that response is as defined in Section 8.4. If the response was an Allocate Response, and the MAPPED-ADDRESS contains the same transport address as previously obtained, the binding has been refreshed. The LIFETIME attribute indicates the amount of additional time the binding will live without activity. If, however, the response was an Allocate Error Response with an ERROR-CODE indicating a 430 response, it means that the binding has expired at the server. The client MAY use the procedures in Section 8.3 to obtain a new binding (this will require a new one-time username and password. Other response codes do not imply that the binding has been expired, just that the refresh has failed.

8.6 Sending Encapsulated Data

Before receiving any UDP or TCP data, a client has to send first. To do that, it uses the Send Request. For UDP, a client MAY send this request at any time. For TCP, it MUST NOT send it after it has transmitted a Set Active Destination Request that yielded a successful result.

The Send request MUST contain a transaction ID, unique for each request, and uniformly and randomly distributed between 0 and 2**128 - 1. The message type of the request MUST be "Send Request". The length is set as described in Section 11.1 of STUN.

The Send request MUST contain the MAGIC-COOKIE attribute as the first attribute. The client MUST include a USERNAME attribute, containing the same username used in the Allocate request for this binding. The request MUST include a MESSAGE-INTEGRITY attribute as the last attribute. The key is equal to the password used for the Allocate request for this binding. For UDP, the Send Request MUST be sent to the same IP address and port as the Allocate Request, and MUST be sent from the same source IP and port used to send the Allocate request for the binding. For TCP, it MUST be sent over the same connection used for the initial allocation. Rules for retransmissions for Send Requests sent over UDP are identical to those for STUN Binding Requests. Transaction timeouts are identical to those for STUN Binding Requests, independent of the transport

protocol.

The Send Request MUST contain a DESTINATION-ADDRESS attribute, which contains the IP address and port that the data is being sent to.

If the server successfully sends the data, the client will receive a Send Response. Note that, as with responses to Allocate refreshes, the client will need to pick a UDP Send Response (or Send Error Response) out of the packet stream by searching for the MAGIC-COOKIE in each received UDP packet.

If the response is an Send Response, the client MUST check the response for a MESSAGE-INTEGRITY attribute. If not present, the client MUST discard the response. If present, the client computes the HMAC over the response. The key MUST be same as used to compute the MESSAGE-INTEGRITY attribute in the request. If the computed HMAC differs from the one in the response, the client MUST discard the response, and SHOULD alert the user about a possible attack. If the computed HMAC matches the one from the response, processing continues.

If the response is a Send Error Response, it is processed as described in the first two paragraphs of Section 8.4. TCP Send Responses are readily identifiable since application data and TURN cannot be used simultaneously on a connection. Thus, until the Set Active Destination Request is sent, the connection is used exclusively for TURN.

8.7 Receiving a Data Indication

Once a client has allocated a binding and used Send to send data on it, the TURN server will start to accept UDP data and incoming TCP connections from the IP addresses that the client sent or connected to. Prior to the establishment of an active destination, all such data received by the server is forwarded to the client using a DATA-INDICATION message. This message generates no response. It contains two attributes - DATA and REMOTE-ADDRESS. The REMOTE-ADDRESS attribute indicates the source transport address that the request came from. 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.

After an active destination is established, for TCP, no more DATA-INDICATION messages will arrive. For UDP, a DATA-INDICATION message will arrive when data comes from an external client that is not equal to the active destination.

8.8 Setting the Active Destination

Once the client has sent data on a binding, it can activate the 5-tuple between the TURN server and a particular external client. By activating it, it means that non-TURN data sent by the client are forwarded on this 5-tuple, and data from the external client are forwarded to the TURN client without encapsulation in a DATA-INDICATION. For TCP, once a destination is activated, TURN signaling over the TCP connection is no longer possible.

To activate a destination, the client constructs a Set Active Destination Request. This request MUST contain a transaction ID, unique for each request, and uniformly and randomly distributed between 0 and 2**128 - 1. The message type of the request MUST be "Set Active Destination Request". The length is set as described in Section 11.1 of STUN.

The Set Active Destination request MUST contain the MAGIC-COOKIE attribute as the first attribute. The client MUST include a USERNAME attribute, containing the same username used in the Allocate request for this binding. The request MUST include a MESSAGE-INTEGRITY attribute as the last attribute. The key is equal to the password used for the Allocate request for this binding. For UDP, the Set Active Destination Request MUST be sent to the same IP address and port as the Allocate Request, and MUST be sent from the same source IP and port used to send the Allocate request for the binding. For TCP, it MUST be sent over the same connection used for the initial allocation. Rules for retransmissions for Set Active Destination Requests sent over UDP are identical to those for STUN Binding Requests. Transaction timeouts are identical to those for STUN Binding Requests, independent of the transport protocol.

The Set Active Destination Request MUST contain a DESTINATION-ADDRESS attribute, which contains the IP address and port of the external client corresponding to the active 5-tuple.

If the response is an Set Active Destination Response, the client MUST check the response for a MESSAGE-INTEGRITY attribute. If not present, the client MUST discard the response. If present, the client computes the HMAC over the response. The key MUST be same as used to compute the MESSAGE-INTEGRITY attribute in the request. If the computed HMAC differs from the one in the response, the client MUST discard the response, and SHOULD alert the user about a possible attack. If the computed HMAC matches the one from the response, processing continues.

If the response is a Set Active Destination Response, the TURN client MUST start a timer, Ta, and set it to 3 seconds. It MUST set its

state to "transitioning". When the timer fires, the client MUST return to normal state. While transitioning, the client MUST NOT send another Set Active Destination request. Furthermore, the client MUST NOT send data without TURN encapsulation (i.e., using a Send Request) while transitioning.

If the response is a Set Active Destination Error Response, and the ERROR-CODE attribute in the response had a value of 439, it means that the client tried to set the active destination while the server was transitioning. The client SHOULD set a timer for 2 seconds, and when the timer fires, retry. If the client received a 440, it is because the client tried to set an active destination to an unknown external client. The TURN client MAY retry with a different destination.

8.9 Tearing Down a Binding

If a client no longer needs a binding, it SHOULD tear it down. For TCP, this is done by closing the connection. For UDP, this is done by performing a refresh, as described in Section 8.5, but with a LIFETIME attribute indicating a time of 0.

8.10 Receiving and Sending Unencapsulated Data

Once a client has set the active destination, it MUST be prepared to receive data from the socket on which the Allocate Request was sent. For UDP, the client MUST be prepared to disambiguate TURN messages from data for the lifetime of the binding. This disambiguation is done using the MAGIC-COOKIE, as described in Section 7.2.3. For TCP, all subsequent data from the server will be application data, and not TURN data.

Once a destination has been activated, the client MAY send data to its peer by sending data on that same socket. Any UDP packets received by the server are forwarded to the default destination address until that address is changed by a subsequent Set Active Destination command. Similarly, any data sent over the TCP connection are forwarded to the TCP connection to the external client.

9. Protocol Details

This section presents the detailed encoding of the message types, attributes, and response codes which are new to TURN. The general message structure of TURN is identical to STUN [1].

9.1 Message Types

TURN defines ten new Message Types:

```
0x0003  :  Allocate Request
0x0103  :  Allocate Response
0x0113  :  Allocate Error Response
0x0004  :  Send Request
0x0104  :  Send Response
0x0114  :  Send Error Response
0x0115  :  Data Indication
0x0006  :  Set Active Destination Request
0x0106  :  Set Active Destination Response
0x0116  :  Set Active Destination Error Response
```

9.2 Message Attributes

TURN defines the following message attributes:

```
0x000d: LIFETIME
0x000e: ALTERNATE-SERVER
0x000f: MAGIC-COOKIE
0x0010: BANDWIDTH
0x0011: DESTINATION-ADDRESS
0x0012: REMOTE-ADDRESS
0x0013: DATA
0x0014: NONCE
0x0015: REALM
```

9.2.1 LIFETIME

The lifetime attribute represents the duration for which the server will maintain a binding 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.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Lifetime                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

9.2.2 ALTERNATE-SERVER

The alternate server represents an alternate IP address and port for a different TURN server to try. It is encoded in the same way as MAPPED-ADDRESS.

9.2.3 MAGIC-COOKIE

The MAGIC-COOKIE is used by TURN clients and servers to disambiguate TURN traffic from data traffic. Its value ix 0x72c64bc6.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1|1|1|0|0|1|0|1|1|0|0|0|1|1|0|0|1|0|0|1|0|1|1|1|1|0|0|0|1|1|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

9.2.4 BANDWIDTH

The bandwidth attribute represents the peak bandwidth, measured in kbits per second, that the client expects to use on the binding. The value represents the sum in the receive and send directions.
[[Editors note: Need to define leaky bucket parameters for this.]]

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Bandwidth                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

9.2.5 DESTINATION-ADDRESS

The DESTINATION-ADDRESS is present in Send Requests and Set Active Destination Requests. It specifies the address and port where the data is to be sent. It is encoded in the same way as MAPPED-ADDRESS.

9.2.6 REMOTE-ADDRESS

The REMOTE-ADDRESS is present in Data Indications. It specifies the address and port from which a packet was received. It is encoded in the same way as MAPPED-ADDRESS.

9.2.7 DATA

The DATA attribute is present in Send Requests 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).

9.2.8 NONCE

The NONCE attribute is present in Shared Secret Requests and Shared Secret Error responses. It contains a sequence of qdtext or quoted-pair, which are defined in [6].

9.2.9 REALM

The REALM attribute is present in Shared Secret Requests and Shared Secret Responses. It contains text which meets the grammar for "realm" as described in RFC 3261, and will thus contain a quoted string (including the quotes).

9.2.10 Response Codes

TURN defines the following new response codes:

300 (Try Alternate): The client should contact an alternate server for this request.

434 (Missing Realm): The REALM attribute was not present in the request.

435 (Missing Nonce): The NONCE attribute was not present in the request.

436 (Unknown Username): The USERNAME supplied in the Shared Secret Request is not known in the given REALM.

437 (No Binding): A Send Request was received by the server, but there is no binding in place for the source 5-tuple.

438 (Send Failed): A Send Request was received by the server over TCP, but the server wasn't able to transmit the data to the requested destination.

439 (Transitioning): A Set Active Destination request was received by the server. However, a previous request was sent within the last three seconds, and the server is still transitioning to that active destination. Please repeat the request once three seconds have elapsed.

440 (No Destination): A Set Active Destination request was received by the server. However, the requested destination has not been one corresponding to the destination of a Send Request, and has not been one for which packets or a connection attempt have been received.

441 (Wrong Username): A TURN request was received for an allocated binding, but it did not use the same username and password that were used in the allocation. The client must supply the proper credentials, and if it cannot, it should teardown its binding, allocate a new one time password, and try again.

10. Security Considerations

TURN servers allocate bandwidth and port resources to clients. Therefore, a TURN server requires authentication and authorization of TURN requests. This authentication is provided by a client digest over TLS, which results in the generation of a one-time password that is used in a single subsequent Allocate Request. This mechanism protects against eavesdropping attacks and man-in-the-middle attacks. The usage of one-time passwords ensures that the Allocate Requests, which 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, but each requires a new one-time username and password. It is RECOMMENDED that servers implement a cap on the number of one-time passwords that are allocated to any specific user at a time (around 5 or 10 should be sufficient). This will prevent floods of Allocate requests from a single user, in an attempt to use up the resources of the system. A single malicious user could generate a single Allocate Request, obtain a binding, and then flood the server with data over this binding, in an attempt to deny others service. However, this attack requires the attacker themselves to receive the data being sent at the server. To ameliorate these kinds of attacks, servers SHOULD implement a bandwidth cap on each binding (conveyed to the client in the BANDWIDTH attribute of the Allocate Response), and discard packets beyond the threshold.

A client will use the transport address learned from the MAPPED-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 TLS and HMAC-based authentication and integrity checks provided in TURN. They can guarantee the authenticity and integrity of the mapped addresssses. 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 STUN. These attacks are based on the fact that a STUN server mirrors the source IP address, which cannot be

authenticated. TURN does not use the source address of the Allocate Request, 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 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 TURN does not appear to be that important, and therefore, this capability is not provided.

TURN 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.

TCP transport addresses allocated by TURN will properly work with TLS and SSL. However, any addresses allocated by TURN will not operate properly with IPSec Authentication Header (AH) [10] in transport mode. IPSec ESP [11] and any tunnel-mode ESP or AH should still operate.

11. 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 [12]. TURN 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. This section meets those requirements.

11.1 Problem Definition

From RFC 3424 [12], any UNSAF proposal must provide:

> Precise definition of a specific, limited-scope problem that is to be solved with the UNSAF proposal. A short term fix should not be generalized to solve other problems; this is why "short term fixes usually aren't".

The specific problem being solved by TURN is for a client, which may be located behind a NAT of any type, to obtain an IP address and port on the public Internet, useful for applications that require a client to place a transport address into a protocol message, with the expectation that the client will be able to receive packets from a single host that will send to this address. Both UDP and TCP are addressed. It is also possible to send packets so that the recipient sees a source address equal to the allocated address. TURN, by design, does not allow a client to run a server (such as a web or SMTP server) using a TURN address. TURN is useful even when NAT is not present, to provide anonymity services.

11.2 Exit Strategy

From [12], any UNSAF proposal must provide:

> Description of an exit strategy/transition plan. The better short term fixes are the ones that will naturally see less and less use as the appropriate technology is deployed.

It is expected that TURN will be useful indefinitely, to provide anonymity services. When used to facilitate NAT traversal, TURN does not iself provide an exit strategy. That is provided by the Interactive Connectivity Establishment (ICE) [13] mechanism. ICE allows two cooperating clients to interactively determine the best addresses to use when communicating. ICE uses TURN-allocated addresses as a last resort, only when no other means of connectivity exists. As a result, as NATs phase out, and as IPv6 is deployed, ICE will increasingly use other addresses (host local addresses). Therefore, clients will allocate TURN addresses, but not use them, and therefore, de-allocate them. Servers will see a decrease in usage. Once a provider sees that its TURN servers are not being used at all (that is, no media flows through them), they can simply remove them. ICE will operate without TURN-allocated addresses.

11.3 Brittleness Introduced by TURN

From [12], any UNSAF proposal must provide:

> Discussion of specific issues that may render systems more "brittle". For example, approaches that involve using data at multiple network layers create more dependencies, increase debugging challenges, and make it harder to transition.

TURN introduces brittleness in a few ways. First, it adds another server element to any system, which adds another point of failure. TURN requires clients to demultiplex TURN packets and data based on hunting for a MAGIC-COOKIE in the TURN messages. It is possible (with extremely small probabilities) that this cookie could appear within a data stream, resulting in mis-classification. That might introduce errors into the data stream (they would appear as lost packets), and also result in loss of a binding. TURN relies on any NAT bindings existing for the duration of the bindings held by the TURN server. Neither the client nor the TURN server have a way of reliably determining this lifetime (STUN can provide a means, but it is heuristic in nature and not reliable). Therefore, if there is no activity on an address learned from TURN for some period, the address might become useless spontaneously.

TURN will result in potentially significant increases in packet latencies, and also increases in packet loss probabilities. That is because it introduces an intermediary on the path of a packet from point A to B, whose location is determined by application-layer processing, not underlying routing topologies. Therefore, a packet sent from one user on a LAN to another on the same LAN may do a trip around the world before arriving. When combined with ICE, some of the most problematic cases are avoided (such as this example) by avoiding the usage of TURN addresses. However, when used, this problem will exist.

Note that TURN does not suffer from many of the points of brittleness introduced by STUN. TURN will work with all existing NAT types known at the time of writing, and for the forseeable future. TURN does not introduce any topological constraints. TURN does not rely on any heuristics for NAT type classification.

11.4 Requirements for a Long Term Solution

From [12]}, any UNSAF proposal must provide:

> Identify requirements for longer term, sound technical solutions -- contribute to the process of finding the right longer term solution.

Our experience with TURN continues to validate our belief in the requirements outlined in Section 14.4 of STUN.

11.5 Issues with Existing NAPT Boxes

From [12], any UNSAF proposal must provide:

Discussion of the impact of the noted practical issues with existing, deployed NA[P]Ts and experience reports.

A number of NAT boxes are now being deployed into the market which try and provide "generic" ALG functionality. These generic ALGs hunt for IP addresses, either in text or binary form within a packet, and rewrite them if they match a binding. This will interfere with proper operation of any UNSAF mechanism, including TURN. However, if a NAT tries to modify a MAPPED-ADDRESS in a TURN Allocate Response, this will be detected by the client as an attack.

12. Examples

TODO.

13. Acknowledgements

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

14. References

14.1 Normative References

[1] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

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

[3] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[4] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

14.2 Informative References

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

[6] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[7] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[8] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[9] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002.

[10] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[11] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

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

[13] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Multimedia Session Establishment Protocols", draft-ietf-mmusic-ice-04 (work in progress), February 2005.

[14] Audet, F. and C. Jennings, "NAT Behavioral Requirements for Unicast UDP", draft-ietf-behave-nat-udp-02 (work in progress), June 2005.

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
Airspace

Email: rohan@ekabal.com

Christian Huitema
Microsoft
One Microsoft Way
Redmond, WA 98052-6399
US

Email: huitema@microsoft.com

Acknowledgment

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