MIDCOM
Internet-Draft
Expires: August 16, 2004

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

February 16, 2004

Traversal Using Relay NAT (TURN)
draft-rosenberg-midcom-turn-04

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'' the tables so that the element on the

inside can be on the receiving end, rather than the sending end, of a connection that is requested by the client.

Table of Contents

1. Introduction 4
2. Terminology 5
3. Definitions 6
4. Applicability Statement 7
5. Overview of Operation 8
6. Message Overview 10
7. Server Behavior 11
7.1 Shared Secret Request 11
7.2 Allocate Request 13
7.2.1 Overview 13
7.2.2 Initial Requests 13
7.2.3 Requests for Pre-Allocated Ports 17
7.2.4 Subsequent Requests 18
7.3 Send Request 19
7.4 Receiving Packets and Connections 20
7.5 Lifetime Expiration 22
8. Client Behavior 23
8.1 Discovery 23
8.2 Obtaining a One Time Password 23
8.3 Allocating a Binding 24
8.4 Processing Allocate Responses 25
8.5 Allocating a Pre-Allocated Binding 26
8.6 Refreshing a Binding 27
8.7 Sending Data 27
8.8 Tearing Down a Binding 28
8.9 Receiving and Sending Data 28
9. Protocol Details 30
9.1 Message Types 30
9.2 Message Attributes 30
9.2.1 TRANSPORT-PREFERENCES 30
9.2.2 LIFETIME 31
9.2.3 ALTERNATE-SERVER 31
9.2.4 MAGIC-COOKIE 31
9.2.5 BANDWIDTH 32
9.2.6 DESTINATION-ADDRESS 32
9.2.7 SOURCE-ADDRESS 32
9.2.8 DATA 32

9.3 Response Codes 32
10. Security Considerations 34
11. IAB Considerations 36
11.1 Problem Definition 36
11.2 Exit Strategy 36
11.3 Brittleness Introduced by TURN 37

11.4 Requirements for a Long Term Solution 38
11.5 Issues with Existing NAPT Boxes 38
12. Examples 39
Normative References 40
Informative References 41
Authors' Addresses 41
Intellectual Property and Copyright Statements 43

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 STUN 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. Once a host sends packets to that transport address, it is ``locked down'', meaning that the client cannot cause packets to be sent to that host through the relay. The client will still receive packets sent from different peers to that transport address, but these are wrapped in TURN protocol headers, reducing their efficiency. This is done purposefully, so as to prevent TURN from being used to run servers (such as a web server or DNS server) on a client behind a NAT. In this way, enterprises which deploy NATs and firewalls to prevent users from running servers, can be confident that TURN will not cause any violations in their enterprise security policies.

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.

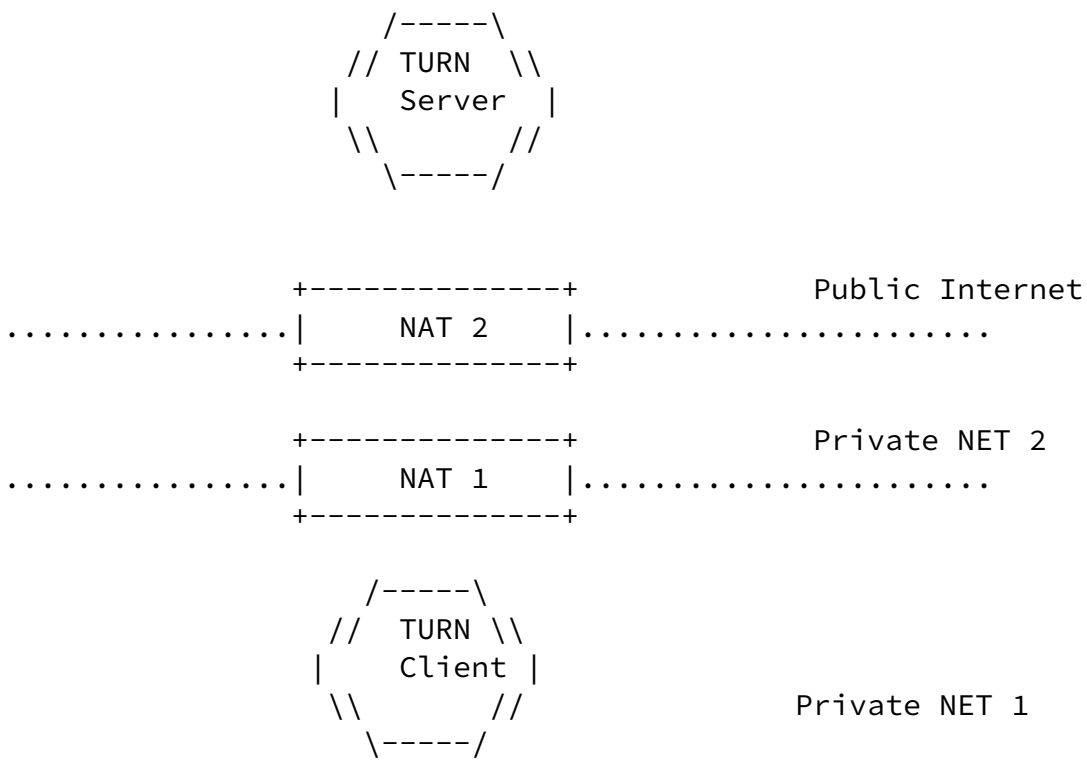


Figure 1

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 for 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 remembers the source transport address that the request came from (call this SA), and returns a public

transport address, PA, in the TURN response. The TURN server is responsible for guaranteeing that packets sent to PA route to the TURN server. The TURN server then waits for data on PA. When data is received (either a UDP packet or a TCP connection request), the TURN server accepts the connection (in the case of TCP), and then stores the remote address and port where the data came from (RA). The data just received, if any, are then forwarded to SA. The TURN server then acts as a relay. Any data received from SA are forwarded to RA. Any data sent from RA to PA are sent to SA. If some other host sends packets to PA, those packets are forwarded to PA as well, but they are sent as a TURN message from the server to the client. This affords some protection against denial of service attacks that would otherwise be possible. TURN also allows a client to send packets through the TURN server before lockdown has occurred, by using the SEND command.

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.

TURN also allows a client to request an odd or even port when one is allocated, and for it to pre-allocate the next higher port. This is useful for securing consecutive ports for usage with the Real Time Transport Protocol (RTP) [5].

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 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 TRANSPORT-PREFERENCES attribute, which allows a client to request an odd or even port, and to pre-allocate the next higher port. It defines the LIFETIME attribute, which allows the TURN server to tell the client when the binding will be released. 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. Finally, it defines the ALTERNATE-SERVER attribute, which allows the server to redirect the TURN client to connect to an alternate server.

7. Server Behavior

The server behavior depends on whether the request is a Shared Secret Request or an Allocate 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. When the server receives packets on this allocated address, it acts as a relay, and forwards them towards the source of the Allocate request. The server remembers the source transport address where that packet came from, and "locks down". This means that packets sent from the client to the TURN server are forwarded to that address.

As a result, the server maintains a set of bindings. These bindings are associations between the five-tuple of received Allocate requests (source IP address and port, destination IP address and port, and protocol), called the allocate five-tuple, and another five tuple, called the remote five-tuple.

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 received with a source transport address which is not associated with any existing bindings. A subsequent request is one received that is associated with an existing binding.

7.2.2 Initial Requests

A TURN server MUST be prepared to receive Binding 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 so, the server

looks for the presence of the TRANSPORT-PREFERENCES attribute in the request. If the attribute indicates that an even port is requested, the server attempts to allocate a transport address with an even port. If the attribute indicates that an odd port is requested, the server attempts to allocate a transport address with an odd port. If the attribute indicates that there is no preference for port parity, or if the TRANSPORT-PREFERENCES attribute was absent, the server attempts to allocate a port with either parity. The server MUST NOT allocate ports from the well-known port range (0-1023) and MUST NOT allocate ports from the user registered port range (1024 through 49151).

> This aspect of the protocol helps guarantee that users cannot run servers (such as a web server, SIP server, or SMTP server) using TURN.

The TRANSPORT-PREFERENCES attribute can also indicate a preference for a specific address and port, pre-allocated previously by a prior Allocate request. This case is described in Section 7.2.3.

If a port meeting the constraints (including bandwidth) 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.

Assuming a port was allocated according to the preferences (call this the base port), the server checks to see if the TRANSPORT-PREFERENCES attribute is present, and indicates a desire to pre-allocate the next higher port (called the pre-allocated port). If so, the server attempts to allocate that port from its local operating system. If it cannot be allocated, the server can do one of two things. First, it MAY try to allocate a different base port, in the hopes that the next higher port is available. If the server believes that there are no adjacent ports meeting the parity constraints present in the request, the server MAY generate an 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 a base port is allocated, the server creates a binding for it. This binding is a mapping between two five-tuples - the allocate five-tuple and the remote five-tuple. The allocate five-tuple is set to the five-tuple of the Allocate Request (that is, the protocol of the allocate five-tuple is set to the protocol of the Allocate Request (TCP or UDP), the source IP address and port of the allocate five-tuple are set to the source IP address and port in the Allocate

Request, and the destination IP address and port of the allocate five-tuple are set to the destination IP address and port in the Allocate Request). The protocol in the remote five-tuple is set to the protocol from the Allocate Request. The source IP address of the remote five-tuple is set to the interface from which the base port was allocated. The source port of the remote five-tuple is set to the base port. If the binding was allocated for TCP, the connection on which the Allocate request was received is associated with the allocate five-tuple in the binding.

The server MUST remember the one-time username and password used to obtain the binding.

If an address and port was pre-allocated (either at the request or the user, or the at the discretion of the server), a binding is also created for it. The allocate five-tuple is left empty. The protocol in the remote five-tuple is set to the protocol from the Allocate Request. The source IP address of the remote five-tuple is set to the interface from which the pre-allocated port was allocated. The source port of the remote five-tuple is set to the pre-allocated port. The identity of the user (defined as the username provided in the Shared Secret Request used to obtain the one-time password used in the Allocate Request) is associated with this pre-allocated tuple. Only that user can perform an allocation for this tuple. Furthermore, a timer is set. If no allocation is made against this pre-allocation within 5 minutes, the port is released and the binding is deleted.

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 cae, the resulting duration is added to the current time, and a timer is set to fire at or after that time. Section 7.5 discusses behavior when the timer fires.

Once the base port has been obtained from the operating system, the pre-allocated port obtained, and the activity timer started for the base 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 server MUST add a MAPPED-ADDRESS attribute to the Allocate

Response. The IP address component of this attribute MUST be set to the interface from which the base port was allocated. The port

component of this attribute MUST be set to the base port.

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. Once the response is sent, the TURN server begins acting as a relay for that connection (see Section 7.4). 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).

Additionally, if the base port was for UDP, the server MUST be prepared to receive UDP packets once the TURN response is sent. If the base port was for TCP, the server MUST be prepared to receive a TCP connection request on that port. Behavior when either occurs is described in Section 7.4.

7.2.3 Requests for Pre-Allocated Ports

The TRANSPORT-PREFERENCES attribute of the Allocate Request can indicate a desire to allocate a port that was previously pre-allocated by a prior Allocate request. If such an indication is present, the server checks that this address and port has been pre-allocated by a previous Allocate Request. The only user authorized to allocate a pre-allocated address is the same one that

generated the pre-allocation. Note that the one-time usernames for both requests (the pre-allocation and the final allocation) will be different. However, both MUST have been obtained through Shared Secret Requests authenticated as being sent from the same user.

If the Allocate request arrives on a different protocol than was used to make the pre-allocation, the server MUST send an Allocate Error Response. That response MUST contain an ERROR-CODE attribute with a response code of 400.

Assuming the requested port has been pre-allocated by the same user, the server completes the allocation by setting the allocate five-tuple for the binding to be equal to that of the Allocate request. The server sets the activity timer for this binding, and generates an Allocate Response. This response MUST contain a MAPPED-ADDRESS attribute which contains the interface from which the pre-allocated port was obtained, along with the pre-allocated port. The response MUST contain a LIFETIME attribute and a MESSAGE-INTEGRITY attribute as well.

7.2.4 Subsequent Requests

Once a binding has been created, non-TURN packets received from the client are generally forwarded to the remote client. However, if the binding is UDP, 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 to allocate that binding previously. That is, the five-tuple from the Allocate request is compared to the allocate five-tuples in existing bindings. The matching binding is selected. The one-time username and password associated with that binding MUST match the ones used in the request.

Any TRANSPORT-PREFERENCE attribute in the request is ignored. An Allocate Request sent to an existing binding is always a refresh or

deallocation. 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 7.5 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 source IP address and port from the remote five-tuple of the binding. 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. 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).

7.3 Send Request

In some networks, enterprise firewall policies prevent users from sending packets directly out to the public Internet. A TURN server can act as a relay for packets sent by a client in such a network. However, the TURN server can only relay packets once the remote five-tuple has been fully filled in with an incoming packet, a process called "locking down". Many applications will require a user to send a packet first in order to trigger such an incoming packet.

These initial packets must also be relayed. To provide this capability, TURN supports the Send Request.

The Send request asks the TURN server to forward a data packet to a specified IP address and port. A Send Request is like any other TURN request. A server can disambiguate a Send Request from a data packet by looking for the MAGIC-COOKIE attribute, as described in Section 7.2.4.

Once the server has identified a request as a Send request, the server verifies that it has arrived with a source five-tuple corresponding to an existing allocation. If there is no matching allocation, the server MUST generate a 437 (No Binding) Send Error Response. If there is a matching allocation, the server checks if the remote 5-tuple for the binding has been filled in (i.e., lock-down has occurred). If it has, the server MUST generate a 438 (Sending Disallowed) 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 to allocate that binding previously. That is, the five-tuple from the Send request is compared to the allocate five-tuples in existing bindings. The matching

binding is selected. The one-time username and password associated with that binding MUST match the ones used in the request.

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. If the value of the port from the DESTINATION-ADDRESS is between 0 and 1023 inclusive, the server MUST reject the request with a 439 (Illegal Port) Send Error Response.

Assuming the Send Request has been validated, the server then takes the contents of the DATA attribute, and creates a UDP packet whose payload equals that content. The server sets the source IP address equal to the source IP from the remote five-tuple, and the source port equal to the source port from the remote five-tuple. The destination address and port are set to the contents of the DESTINATION-ADDRESS. 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.

Once the UDP packet is sent, 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. 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.

7.4 Receiving Packets and Connections

If a TURN server receives a TCP connection request on a port it has allocated, the server retrieves the binding whose remote five-tuple has a source address and source port that match the IP address and port to which the connection was made, and whose transport is TCP. If the destination IP address and port of the remote five-tuple in the binding are already filled in (which means that a connection was already made to this tuple), the connection request is rejected. Otherwise, it is accepted. If the connection is accepted, the server MUST set the destination IP address and port of the remote five-tuple to the source IP address and port in the SYN packet. It also associates this connection with the remote five-tuple.

If a TURN server receives a UDP packet on a port it has allocated,

the server retrieves the binding whose remote five-tuple has a source address and source port that match the IP address and port to which the packet was sent, and whose transport is UDP. If the destination IP address and port of the remote five-tuple in the binding are already filled in, and do not match the source IP address and port of the UDP packet, the server transmits the packet to the client using a Data Indication message. This is a TURN message that 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 DATA attribute

whose contents are equal to the payload of the UDP packet. The message MUST contain a SOURCE-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 client using the allocate five-tuple. That is, its destination address is equal to the source address from the allocate five-tuple, and its source address is equal to the destination address from the allocate five-tuple.

If the packet was not sent as a Data Indication message, it is forwarded. To forward, the packet is sent with a source IP address and port equal to the destination IP address and port in the allocate five-tuple, and with a destination address and port equal to the source IP address and port in the allocate five-tuple. If the destination address and port of the remote five-tuple were not filled in, they are populated at this time. The server MUST set the destination IP address and port of the remote five-tuple to the source IP address and port in the UDP packet. Note that, unlike a Data Indication message, when the packet is forwarded, the payload of the transmitted packet is identical to the one received. No headers are added to the packet.

The process of filling in the destination IP address and port of the remote five-tuple is called "locking down". Once done, the client can only send and receive packets with the specific peer from which the first packet or connection was received.

If a TURN server receives data on a TCP connection that was opened to a port it had allocated, the server MUST forward this data onto the connection associated with allocate-tuple in the binding.

If a TURN server receives data on a TCP connection that is associated with an allocate five-tuple, the binding for that tuple is retrieved. If the destination IP address and port of that tuple have not been filled in yet, the data is discarded. If the destination address and port have been filled in, the connection associated with the remote five-tuple is obtained, and the data is forwarded on that connection.

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

Similarly, if a TURN server receives a UDP packet on one of its

public TURN ports, it checks to see if the source IP address and port match those of the allocate five-tuples in an existing binding. If there is a match, the the UDP packet is not a TURN request (see Section 7.2.4 for details on how this determination is made), the destination IP address and port in the remote five-tuple of the binding are checked. If they are not filled in yet, the UDP packet is discarded. If they are, the packet is forwarded. It is forwarded using the source IP address and port from the remote five-tuple, and a destination IP address and port from the remote five-tuple.

If a TCP connection associated with an allocate five-tuple is closed, the connection associated with the corresponding remote five-tuple is also closed. At that point, the binding is destroyed. Similarly, if the TCP connection associated with a remote five-tuple is closed, the connection associated with the corresponding allocate five-tuple is closed, and the binding is destroyed.

7.5 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.

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 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 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 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 greater than 0x7fff, the attribute MUST be ignored. If the client receives a Shared Secret Response with an attribute whose type is 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. If the client wishes to allocate an odd or even port, it can do so by including the TRANSPORT-PREFERENCES attribute in the request. That attribute can also be used by the client if it wishes to pre-allocate the port one higher.

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. 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 greater than 0x7fff, the attribute MUST be ignored. If the client receives a response with an attribute whose type is 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 Binding 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 Allocating a Pre-Allocated Binding

If the initial Allocate Request included TRANSPORT-PREFERENCES that indicated a desire to pre-allocate the port one-higher, the client MAY allocate that port at a later time. It MUST do so within 4 minutes of receiving the Allocate Response, or the pre-allocated port will expire.

To allocate the port, the client generates an Allocate Request as described in Section 8.3. A new username and password MUST be used for this allocation. The request MUST contain a TRANSPORT-PREFERENCES attribute. It MUST indicate an explicit interface and port, whose value is one higher than the port number returned in the prior Allocate Response.

Processing of the responses is identical to Section 8.4. However, the client SHOULD explicitly check that received packets are TURN responses, as opposed to data packets, using the techniques described

in Section 7.2.4.

8.6 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 7.2.4. 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.

8.7 Sending Data

Before lockdown has occured, a client MAY send data using a binding it has allocated from the TURN server. To do that, it formulates a Send 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 "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. 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. Rules for retransmissions for Send Requests sent over UDP are identical to those for STUN Binding Requests. There is currently no support for Send Requests over TCP. 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. A client MUST NOT specify a port below 1024, as the server will reject such requests. This prevents TURN from being used as a relay to launch DoS attacks against well-known services. The Send Request MUST contain a DATA attribute, whose contents are the data to transmit.

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 the 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 a Send Error Response, it is processed as described in the first two paragraphs of Section 8.4. If the response code is 438, the client is forbidden from using the Send Request, since lockdown has occurred. The client can relay data to the peer by sending the data without a TURN message wrapper. [[OPEN ISSUE: is there a need for the client to be told what the locked-down address is?]]

8.8 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 8.6, but with a LIFETIME attribute indicating a time of 0.

8.9 Receiving and Sending Data

Once a binding has been allocated by an Allocate Response, the client 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 7.2.4.

Once data has been received, the client MAY send data to its peer by sending data on that same socket. Sending data on the socket before data is received will cause the data to be discarded by the server.

The client may receive a Data Indication message from the TURN server. The client does not generate any kind of response to this message. Its receipt implies that a packet from a second peer has been received after lock-down. This specification does not define any particular treatment to data received in such an indication. However, in many cases, it can be a sign of a potential denial-of-service

attack against the client. If the client believes that it should not be receiving data from any other source, it SHOULD terminate the binding.

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 three 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
```

9.2 Message Attributes

TURN defines the following message attributes:

```
0x000c: TRANSPORT-PREFERENCES
0x000d: LIFETIME
0x000e: ALTERNATE-SERVER
0x000f: MAGIC-COOKIE
0x0010: BANDWIDTH
0x0011: DESTINATION-ADDRESS
0x0012: SOURCE-ADDRESS
0x0013: DATA
```

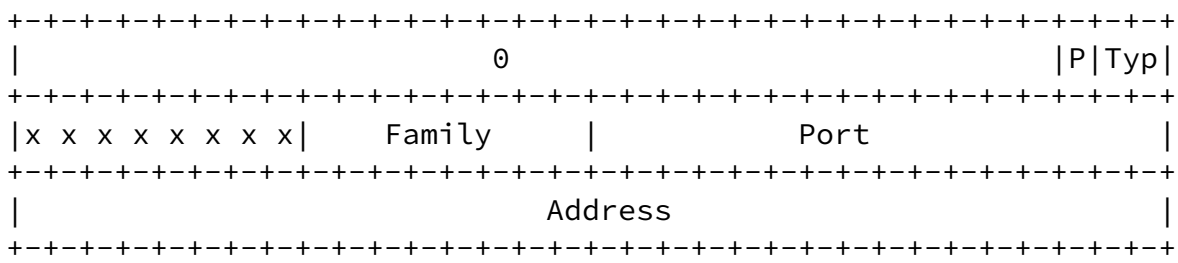
9.2.1 TRANSPORT-PREFERENCES

The TRANSPORT-PREFERENCES attribute indicates preferences for the ports allocated by the TURN server. It is either 32 or 96 bits long, depending on the value of the Typ bits. These bits indicate the preferences for the allocated port:

```
0b00: no preferences
0b01: odd port parity
0b10: even port parity
0b11: allocate a pre-allocated port
```

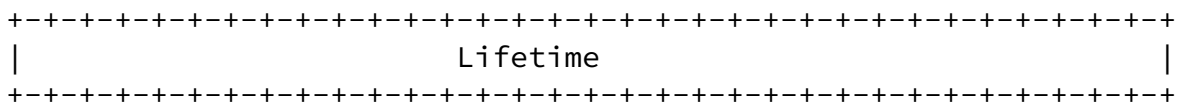
When the Typ bits are 0b11, the following 64 bits encode the pre-allocated transport address. They are in the same format used for MAPPED-ADDRESS.

The P bit indicates a desire for pre-allocating the port one-higher. If 1, it means pre-allocation is desired. This bit MUST NOT be set to 1 if the Typ bits are 0b11. That is, pre-allocation cannot be done again when allocating a previously pre-allocated port.



9.2.2 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.

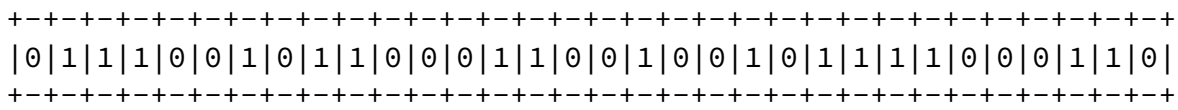


9.2.3 ALTERNATE-SERVER

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.

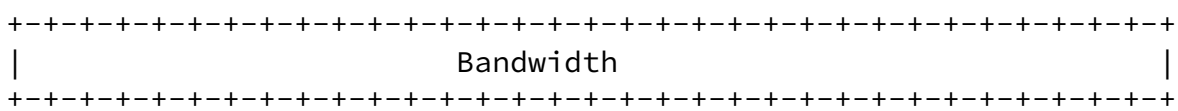
9.2.4 MAGIC-COOKIE

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



9.2.5 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.]]



9.2.6 DESTINATION-ADDRESS

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

9.2.7 SOURCE-ADDRESS

The SOURCE-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.

9.2.8 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).

9.3 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 (Sending Disallowed): A Send Request was received by the server, but lock-down has already occurred, and sending is disallowed.

439 (Illegal Port): A Send Request was received by the server, but lock-down has already occurred, and sending is disallowed.

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 Binding 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 addressses. 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 Binding Request, and therefore, those attacks do not apply.

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.

Once a binding is locked down by the receipt of a packet, a client is prohibited from using the binding to send packets anywhere else. If an eavesdropper had observed a packet containing a TURN allocated

address, it can transmit a packet to this address in an attempt to cause lock-down. This will prohibit a legitimate user from communicating with the client on that address. This particular attack is most easily prevented by ensuring that confidentiality is provided in any protocols that are used to transport a TURN binding. However, in a variant of this attack, a malicious client may flood a TURN server with UDP packets over a wide port range, in an attempt to cause lock-down on any bindings which were just allocated. This attack cannot be prevent with confidentiality mechanisms within other protocols. Fortunately, this attack is expensive to launch. Because the server provides no positive indications of lock-down, an attacker will need to be flooding continously without any indication of success. Furthermore, the rate of packets sent to any particular port needs to be very high - on the order of one every second or so - since there is a limited window of opportunity for locking down before a legitimate client sends a packet to the binding and causes lock-down. This attack can also be detected by clients. They will still receive the legitimate packets through the TURN Data Indications. In many cases, a client will be able to disambiguate the legitimate ones from those from the attacker. If it determines an attack is in progress, it can terminate the binding and retry.

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.

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.

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 Nettwork Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP)", draft-rosenberg-sipping-ice-01 (work in progress), July 2003.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

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

Rohan Mahy
Cisco Systems
101 Cooper St
Santa Cruz, CA 95060
US

EMail: rohan@cisco.com

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

EMail: huitema@microsoft.com

Intellectual Property Statement

IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

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

Full Copyright Statement

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment

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