# DECLARATION

I, Alexa Morris, based on my personal knowledge and information, hereby declare as follows:

1. I am Managing Director of the IETF Administration LLC and have held that position since the LLC was formed in August 2018. Prior to that, starting on January 1, 2008, I was the Executive Director of the Internet Engineering Task Force, which was an activity of the Internet Society. Since the business of IETF did not change in any materially relevant manner with the formation of the LLC, I will collectively refer to both the activity and the LLC as IETF.

2. One of my responsibilities with IETF has been to act as the custodian of Internet-Drafts and records relating to Internet-Drafts. I am familiar with the record keeping practices relating to Internet-Drafts, including the creation and maintenance of such records.

3. I hereby declare that all statements made herein are of my own knowledge and information contained in the business records of IETF and are true, and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements may be punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

4. Since 1998, it has been the regular practice of the IETF to publish Internet-Drafts and make them available to the public on its website at www.ietf.org (the IETF website). The IETF maintains copies of Internet-Drafts in the ordinary course of its regularly conducted activities.

5. Any Internet-Draft published on the IETF website was reasonably accessible to the public and was disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art exercising reasonable diligence could have located it. In particular, the Internet-Drafts were indexed and searchable on the IETF website.

6. Internet-Drafts are posted to an IETF online directory. When an Internet-Draft is published, an announcement of its publication that describes the Internet-Draft is disseminated. Typically, that dated announcement is made within 24 hours of the publication of the Internet-Draft. The announcement is kept in the IETF email archive and the date is affixed automatically.

7. The records of posting the Internet-Drafts in the IETF online repository are kept in the course of the IETF's regularly conducted activity and ordinary course of business. The records are made pursuant to established procedures and are relied upon by the IETF in the performance of its functions.

8. It is the regular practice of the IETF to make and keep the records in the online repository.

9. Exhibit 1 is a true and correct copy of an announcement of the publication of draft-rosenberg-midcom-turn-00.txt, titled "Traversal Using Relay NAT (TURN)". I have determined that an announcement of the publication of this Internet-Draft was made on November 20, 2001. Therefore, based on the normal practice of the IETF, that Internet-Draft was reasonably available to the public within 24 hours of that announcement. At that time, the Internet-Draft would have been disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located it.

Pursuant to Section 1746 of Title 28 of United States Code, I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that the foregoing is based upon personal knowledge and information and is believed to be true.

Date: March 6, 2019 By: [illegible]
Alexa Morris

EXHIBT 1

```
Internet Engineering Task Force                             MIDCOM WG
Internet Draft                        Rosenberg,Weinberger,Huitema,Mahy
draft-rosenberg-midcom-turn-00.txt       dynamicsoft,Microsoft,Cisco
November 14, 2001
Expires: March 2002
```

# Traversal Using Relay NAT (TURN)

STATUS OF THIS MEMO

Abstract

Traversal Using Relay NAT (TURN) is a simple 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.

## 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. Guidlines have been developed [1] that describe how to build "NAT friendly" protocols, but many protocols simply cannot be constructed according to those guidelines. Examples of such protocols include almost all peer-to-peer protocols, for example.

To handle this, we have documented the Simple Traversal of UDP Through NAT (STUN) protocol [2], which allows for clients behind a NAT to discover the presence of the NAT, and then to allocate an address that is useful for receiving data in the case where they are behind a full-cone or restricted-cone NAT. However, it is acknowledged in that draft that while STUN allows a client to discover that its behind a symmetric NAT, it provides no assistance in traversing symmetric NATs.

This protocol serves as a complement to STUN, handling the case where the user is behind a symmetric NAT. It allows a client to request an IP address and port that it can receive data on from any other host on the Internet. This is accomplished using a server in the service provider cloud, known as a TURN server. When a host on the Internet sends to this IP address and port, the TURN server creates an association between the two. The client behind the NAT will receive this, and any other subsequent data from that host. In addition, the client behind the NAT can send data, and that data will be forwarded by the TURN server to the host which connected. TURN servers purposefully support a single association, so that only a single host can be connected using the IP address and port provided by the turn server. This assures that TURN can't be used to violate the policy that symmetric NAT and firewalls are meant to enforce. All TURN does is allow a client to communicate with a single peer whose address it doesn't know ahead of time. TURN is not a tunneling protocol, and therefore does not allow for a user to send and receive UDP, if, for example, the firewall policy prohibits the usage of UDP. Effectively, a TURN server is a NAT function at the UDP and TCP layer, and thus the name of the protocol - its a "relay NAT".

## 2 Do we need this Protocol?

Originally, the TURN protocol was integrated with the STUN protocol documented in [2]. The authors yanked it out of that document because it solves a sufficiently different problem, with differing requirements. We also observed that there are many other potential solutions for the symmetric case, including RSIP [3] [4], and more traditional VPN tunnels. We therefore had to ask ourselves why another solution was needed in this space. Here are some of the

issues we came up with:

- o RSIP and the VPN solutions all allocate an entire IP address to the client. This means the provider must have sufficient IP addresses for all the clients which would simultaneously need service. This could require significant address space. With this proposal, its an IP address/port thats allocated, which means very few IP addresses are needed. This is the standard NAT argument.

- o RSIP and VPN solutions all require tunneling. In this proposal, there is no tunneling. The result is more efficient bandwidth usage, which is important for media packets (RTP is a likely user of this mechanism).

- o RSIP and VPN solutions might contradict enterprise firewall policy, allowing people to run servers, to use UDP when only TCP is allowed, and so on. Some would consider this a feature, not a drawback. But, if the goal is consistency with IT established policies, it is a drawback. Our proposal provides a simple, minimalistic functionality that is consistent with enterprise policy. The only feature TURN adds, is the ability of a user behind the firewall/NAT to receive a single incoming connection, which it has previously requested.

Whether these benefits outweigh the cost of developing and deploying another protocol is important to consider further.

## 3 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALLNOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [5] and indicate requirement levels for compliant STUN implementations.

## 4 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 SIP 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.

## 5 Overview of Operation

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

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


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


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

Figure 1: TURN Configuration

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 is a simple client-server protocol. There is just a single 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 [6]. 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 digest authentication capability, mirroring the operation of HTTP digest [7] that allows the server to authenticate the client, and for the client to authenticate the server. Assuming the request is authenticated, the TURN server remembers the source IP address and port that the request came from (call this SA:SP), and returns a public IP address and port, PA:PP, in the TURN response. This public address and port have to route to the TURN server. The TURN server then waits for data on PA:PP. 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:RP). The data just received, if any, are then forwarded to SA:SP. The TURN server then acts as a relay. Any data received from SA:SP are forwarded to RA:RP. Any data sent from RA:RP to PA:PP are sent to SA:SP. 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 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.

## 6 Message Overview

TURN messages are TLV (type-length-value) encoded using big endian (network ordered) binary. TURN messages are formatted identically to STUN messages, as it is expected that these protocols will frequently be used together.

TURN uses the MAPPED-ADDRESS attribute defined in STUN. This address always appears in the Allocate Response. TURN also defines a CHALLENGE and an AUTHENTICATION attribute. They are very similar to the Authorization and WWW-Authenticate headers in RFC 2617 [7], and convey a realm, nonce, username, and signature. Unlike HTTP Digest, TURN authentication covers the entire message.

A LIFETIME attribute indicates how long the mapped address in the

Allocate response is valid for. The ALTERNATE-SERVER attribute in an Allocate response indicates that the allocation server was full, and the alternate should be used instead.

## 7 Server Behavior

A TURN server generates a single response when a request is received (assuming the request is not discarded). The response MUST contain the same transaction ID contained in the request. The length in the message header MUST contain the total length of the message in bytes, excluding the header.

### 7.1 Client Authentication

The request can be authenticated. This is done using a challenge-response mechanism. When a request is received without proper credentials (which are present in the AUTHENTICATION attribute), the server MAY generate a challenge response. A challenge response MUST NOT contain any attributes except the CHALLENGE attribute. This attribute contains a realm and a nonce. The usage of the realm and nonce is identical to their usage in responses for Digest authentication to HTTP requests, as described in RFC 2617 [7].

The client, upon receiving this challenge, can generate a new request, this time with an AUTHENTICATION attribute, which reflects the nonce and realm back to the server, and contains a keyed hash over the message using the user's name and password. When this is received at the server, the server validates the AUTHENTICATION attribute. This is done by computing the keyed hash in the same way the client does, and comparing the results. If there is a match, the server considers the request authenticated. Otherwise, if it fails, the server SHOULD proceed as if the AUTHENTICATION attribute where not present, typically resulting in another challenge.

## 7.2 Server Authentication

The client can demand authentication of the server. To do this, it includes a CHALLENGE attribute in the request. When the server receives this, assuming that the server has authenticated the request, the response contains (in addition to the other attributes) an AUTHENTICATION attribute. A TURN Server MUST NOT include an AUTHENTICATION attribute in a response, if the server did not successfully authenticate the client in the corresponding request. This attribute contains a hash of the message contents, the nonce, and the shared secret between the client and server.

## 7.3 Allocation

Allocation 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.

When a client is behind a symmetric NAT, the IP address and port it obtains from the Allocate response cannot be used to receive packets from any host on the Internet. Only the recipient of the request can send packets to the client at the mapped address. Unfortunately, this is therefore limited to the server that received the TURN request. Therefore, the server acts as an intermediary. It returns its own IP address and a free port in the response. This can be used by the client in any applications it is running. Any packets received by the server on that IP address and port are forwarded to the client. Since the server is on the public Internet and not natted, anyone can send to it.

As a result, the server MUST maintain a set of mappings. These mappings 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.

When an authenticated Allocate request is received, a partially filled-in remote five-tuple is constructed. This remote five-tuple consists of the same protocol as the five-tuple from the Allocate request, and a destination IP address and port that route to the TURN server. This IP address and port from the remote five-tuple is known as the mapped address. The mapped address is returned to the client in the TURN response, using the MAPPED-ADDRESS attribute. The address and port in the MAPPED-ADDRESS attribute MUST NOT correspond to an address and port already present in another remote five-tuple. Effectively, it is a new address and port that is allocated to the client, and thus the name of the request. Of course, it is possible that there are no more address/port pairs available, due to depleted resources. In that case, the server SHOULD generate a response that

MUST NOT contain a MAPPED-ADDRESS attribute. Instead, it MAY contain an ALTERNATE-SERVER attribute, which contains the address and port of an alternate server. If the ALTERNATE-SERVER attribute is not present, the client will instead use DNS procedures, described below, to find an alternate.

The TURN server MUST listen for packets on the MAPPED-ADDRESS, using the protocol in the remote five-tuple. When a packet is received, the source IP address and port of that packet MUST be used to fill in the remaining two fields in the remote five-tuple. In the case of TCP, the TURN server MUST accept the TCP connection. In the case of UDP, any data present in the packet MUST be forwarded to the source address and port of the allocate five-tuple, and MUST be sent from

the destination address and port of the allocate five-tuple, using the protocol of the five-tuple.

From then on, any packets received on the MAPPED-ADDRESS, with a source IP and port matching the source IP and port of the remote tuple, MUST have their data forwarded to the allocate five-tuple in the same fashion described above. In the case of TCP, any other connection requests to the MAPPED-ADDRESS MUST be refused.

In the case of TCP, if either connection (the one associated with the allocate five-tupe or the one associated with the remote five-tuple) is closed, the TURN server MUST close the other connection, and destroy the mapping between the tuples.

The TURN server SHOULD maintain an activity timer for the mapping. This timer fires after a configurable amount of time (called the lifetime) has expired without data having been received from either five-tuple. When this timer fires, both connections are closed (in the TCP case), and the mapping between the tuples MUST be destroyed. If the TURN server is using activity timers, it MUST include the lifetime interval in the LIFETIME attribute of the original Allocate request.

> OPEN ISSUE: Might be nice to request a lifetime in the Allocate request. TURN could be used for very long lived associations, such as a connection between a user and its proxy. Asking for a long one in that case (only useful for TCP) would be a good thing.

If the TURN server receives data on the Allocate tuple before the remote-tuple has been filled in, the TURN server MUST treat that data as a TURN request. This means it will be responded to as the original request, providing the same MAPPED-ADDRESS once more. This is needed for reliability purposes.

## 8 Client Behavior

The behavior of the client is very simple. Its main task is to discover the TURN server, formulate the request, and handle request reliability.

## 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 specified in [6]. The service name is "turn". The protocol can be either "udp" or "tcp".

> There is no reason at all that a turn server couldn't also make use of SCTP.

The procedures of RFC 2782 are followed to determine the server to contact, with the following additions. If an attempt is made to contact a server, and that attempt results in an ICMP error, or no response with 30 seconds, the client SHOULD attempt to contact the next server. Furthermore, if the client is trying to contact a server, and a server was contacted, but the response did not contain a MAPPED-ADDRESS or ALTERNATE-SERVER attribute, the client SHOULD attempt to contact the next server.

The default port for TURN requests is [to be assigned by IANA]. Administrators SHOULD use this port in their SRV records, but MAY use others.

> This would allow a firewall admin to open the TURN port, so hosts within the enterprise could access new applications. Whether they will or won't do this is a good question.

## 8.2 Authentication

A request formulated by the client follows the syntax rules defined in Section 10. Any two requests that are not bit-wise identical, or not sent to the same server from the same IP address and port, MUST carry different transaction IDs. The transaction ID MUST be uniformly and randomly chosen between 0 and 2^^32 - 1.

Once formulated, the client sends the request. Reliability is accomplished through client retransmissions. Clients SHOULD retransmit the request starting with an interval of 100ms, doubling every retransmit. The client MAY give up after 32 seconds, or MAY continue trying.

If the response contains a CHALLENGE attribute, the client formulates a new request (with a new transaction ID), but otherwise identical to the previous request, with the addition of the AUTHENTICATION attribute. The realm and nonce fields of this attribute are copied from the response. The username is the user's identity at the given realm. The signature is computed as described in Section 10.2.2.

A request with an AUTHENTICATION attribute MAY also contain a CHALLENGE attribute, requesting authentication of the server.

A client MAY cache the realm and nonce fields from the response, and use them to construct the AUTHENTICATION attribute in subsequent requests to the same TURN server (identified by the destination IP address and port).

## 8.3 Allocate request

An Allocate request has no mandatory attributes, and the only optional attributes are AUTHENTICATION and CHALLENGE, whose usage is described above.

If the response contains an ALTERNATE-SERVER attribute, the client SHOULD formulate a new Allocate request, and send it to that server. Otherwise, if there is no ALTERNATE-SERVER attribute, but no MAPPED-ADDRESS attribute, the client SHOULD continue SRV procedures from the point it left off to find the next available server.

Otherwise, the response will contain a MAPPED-ADDRESS attribute with an IP address and port that the client can use within an application. The response will also contain a LIFETIME attribute, which indicates amount of time until the mapping will be invalidated.

The TURN client should listen for data on the same socket used to send the Allocate address. Any data sent to the MAPPED-ADDRESS will show up on this socket. Once it receives data, the client can send data, and it will be delivered to the same host and port which sent the data to the MAPPED-ADDRESS.

## 9 Example Usage

### 9.1 UDP Allocation

Figure 2 shows the process of allocating a request for receipt of UDP packets.

In message 1, the client sends a TURN Allocate request to the server. This passes through the NAT, which rewrites the source address (message 2). The TURN server allocates a MAPPED-ADDRESS, 9.8.7.6:1124, and returns it in the TURN response (message 3). This response has its destination rewritten by the NAT (message 4). The client can then use this information in an application, such as SIP [8], and the result is that the address is passed to some other element (message 5), called the peer. The peer then decides to send data of some sort (perhaps RTP packets), to the client. It sends it to the mapped address, 9.8.7.6:1124, which will arrive at the TURN server and then is forwarded to the client. Any data the client sends is then forwarded back to the peer.

### 9.2 Authentication

```
  TURN       NAT       Turn
  Client               Server
     |          |          |
     | Allocate            |
     | with CHALLENGE      |
     |-------->|--------->|
     |          |          |
     | Response with       |
     | CHALLENGE           |
     |<--------|<---------|
     |          |          |
     | Allocate with       |
     | CHALLENGE and       |
     | AUTHENTICATION      |
     |-------->|--------->|
     |          |          | <-- TURN Server now waiting
     | Response with       |     on MAPPED-ADDRESS
     | AUTHENTICATION and  |
     | MAPPED-ADDRESS      |
     |          |  X <-----|     Packet Loss
     |          |          |
     | .. client waits ..  |
     |          |          |
     | Allocate with       |
     | CHALLENGE and       |
     | AUTHENTICATION      |
     |-------->|--------->|
     |          |          |
     | Response with       |
     | AUTHENTICATION and  |
     | MAPPED-ADDRESS      |
     |<--------|----------|
     |          |          |
```

Figure 3: Flow for mutual authentication

Figure 3 shows the basic flow for mutual authentication. The client sends a request with a challenge. The server wishes to authenticate the client, so it responds to the request with its own challenge, but no authentication attribute. The client retries the request, once again with a challenge attribute and also with an authentication attribute. The server accepts this, and sends a response with its own authentication attribute, along with the mapped address. A retransmit

```
      Client                   NAT              Turn Server           Peer
         |(1) Allocate          |                   |                   |
         | src=10.0.1.1:8898    |                   |                   |
         | dest=9.8.7.6:8765    |(2) Allocate       |                   |
         |--------------------->| src=1.2.3.4:6544  |                   |
     |                    | dest=9.8.7.6:8765 |                    |
     |                    |------------------>|                    |
     |                    |(3) Response       |                    |
     |(4) Response        | mapped=9.8.7.6:1124, src=9.8.7.6:8765  |
     | mapped=9.8.7.6:1123 dest=1.2.3.4:6544  |                    |
     | src=9.8.7.6:8765   |<------------------|                    |
     | dest=10.0.1.1:8898 |                   |                    |
     |<-------------------|                   |                    |
     |                    |                   |                    |
     |(5) Sends 9.8.7.6:1124 to peer          |                    |
     |-------------------------------------------------------------->|
     |                    |                   |                    |
     |                    |                   |(6) Data            |
     |                    |                   | src=5.5.5.5:5555   |
     |                    |(7) Data           | dest=9.8.7.6:1124  |
     |                    | src=9.8.7.6:8765  |<-------------------|
     |(8) Data            | dest=1.2.3.4:6544 |                    |
     | src=9.8.7.6:8765   |<------------------|                    |
     | dest=10.0.1.1:8898 |                   |                    |
     |<-------------------|                   |                    |
     |                    |                   |                    |
     |(9) Data Response   |                   |                    |
     | src=10.0.1.1:8898  |                   |                    |
     | dest=9.8.7.6:8765  |(10) Data Response |                    |
     |------------------->| src=1.2.3.4:6544  |(11) Data Response  |
     |                    | dest=9.8.7.6:8765 | src=9.8.7.6:1124   |
     |                    |------------------>| dest=5.5.5.5:5555  |
     |                    |                   |------------------->|
     |                    |                   |                    |
```

Figure 2: Example flow

of the request triggers the same response to be sent.

## 10 Protocol Details

This section presents the detailed encoding of the attributes which are new to TURN. The general message structure is identical to STUN [2].

### 10.1 Message Header

TURN defines two new Message Types:

0x0002  :  Allocate Request

0x0102  :  Allocate Response

## 10.2 Message Attributes

The following additional attributes are defined:

```
0x0004: CHALLENGE
0x0005: AUTHENTICATION
0x0006: LIFETIME
0x0007: ALTERNATE-SERVER
```

### 10.2.1 CHALLENGE

The CHALLENGE attribute contains a challenge, either from the server, for credentials in order to process the request, or from the client, for credentials in order to process the response.

The CHALLENGE contains two strings: a realm, and a nonce. Both are encoded using a 16 bit length followed by the string. The string MUST NOT be null terminated. The 32 bit alignment of the lengths in the diagram below is for readability purposes only. No padding is required after the end of the string for the realm.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Length                |         Realm               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Length                |         Nonce               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The realm represents a domain over which the entity is to supply a username and password. It is defined in [7]. The nonce is a randomly chosen string that is fed into the signature computation. Nonce selection procedures can be found in [7] and [9].

### 10.2.2 AUTHENTICATION

The authentication attribute provides credentials. It contains three strings: a realm, a nonce, a signature, and a username. They are encoded using a 16 bit length, followed by the string (the strings MUST NOT be null terminated).

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Length                |         Realm               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Length                |         Nonce               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Length                |         Signature           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Length                |         Username            ...
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The realm and nonce are described in [7]. The username is the user identity. The signature is computed as follows.

The entire TURN request, including the TURN headers, up to the end of the last attribute before the AUTHENTICATION attribute, is taken as string "S". String "S" is base64 encoded to become string "B". The signature is computed as the request-digest token, according to the rules of RFC 2617, as if A1 was equal to string "B", and qop was unspecified.

### 10.2.3 LIFETIME

The lifetime attribute represents the duration that a mapping is valid. It is a 32 bit value representing the number of seconds remaining until expiration.

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

### 10.2.4 ALTERNATE-SERVER

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

## 11 Security Considerations

TURN servers, unlike STUN servers, create state upon processing of requests. As a result, they SHOULD authenticate all requests before allocating a mapping to the client. Furthermore, it is RECOMMENDED that authorization policies be used to prevent a single user from allocating more than a configured number of mappings. This prevents hogging of resources by an attacker.

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.

TURN has the important property that compromise of the TURN servers cannot cause security breaches when the client is within an enterprise. The only thing that a compromised server can do is return false addresses, resulting in the inability of the client to receive any data at all. The protocol is therefore fail safe.

## 12 Authors Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Joel Weinberger
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jweinberger@dynamicsoft.com

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

email: huitema@microsoft.com

Rohan Mahy
Cisco Systems
170 West Tasman Dr, MS: SJC-21/3
Phone: +1 408 526 8570
Email: rohan@cisco.com

## 13 Bibliography

[1] D. Senie, "NAT friendly application design guidelines," Internet Draft, Internet Engineering Task Force, Mar. 2001. Work in progress.

[2] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - simple traversal of UDP through NATs," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[3] M. Borella, J. Lo, D. Grabelsky, and G. Montenegro, "Realm specific IP: framework," Request for Comments 3102, Internet Engineering Task Force, Oct. 2001.

[4] M. Borella, D. Grabelsky, J. Lo, and K. Taniguchi, "Realm specific IP: protocol specification," Request for Comments 3103, Internet Engineering Task Force, Oct. 2001.

[5] S. Bradner, "Key words for use in RFCs to indicate requirement levels," Request for Comments 2119, Internet Engineering Task Force, Mar. 1997.

[6] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," Request for Comments 2782,

Internet Engineering Task Force, Feb. 2000.

[7] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, "HTTP authentication: Basic and digest access authentication," Request for Comments 2617, Internet Engineering Task Force, June 1999.

[8] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[9] J. Rosenberg, "Request header integrity for SIP and HTTP digest using predictive nonces," Internet Draft, Internet Engineering Task

Force, June 2001. Work in progress.

Table of Contents

| | | |
|---|---|---|
| 1 | Introduction | 1 |
| 2 | Do we need this Protocol? | 2 |
| 3 | Terminology | 3 |
| 4 | Definitions | 3 |
| 5 | Overview of Operation | 4 |
| 6 | Message Overview | 5 |
| 7 | Server Behavior | 6 |
| 7.1 | Client Authentication | 6 |
| 7.2 | Server Authentication | 6 |
| 7.3 | Allocation | 6 |
| 8 | Client Behavior | 8 |
| 8.1 | Discovery | 8 |
| 8.2 | Authentication | 9 |
| 8.3 | Allocate request | 10 |
| 9 | Example Usage | 10 |
| 9.1 | UDP Allocation | 10 |
| 9.2 | Authentication | 10 |
| 10 | Protocol Details | 13 |
| 10.1 | Message Header | 13 |
| 10.2 | Message Attributes | 13 |
| 10.2.1 | CHALLENGE | 13 |
| 10.2.2 | AUTHENTICATION | 14 |
| 10.2.3 | LIFETIME | 14 |
| 10.2.4 | ALTERNATE-SERVER | 15 |
| 11 | Security Considerations | 15 |
| 12 | Authors Addresses | 15 |
| 13 | Bibliography | 16 |

□