TRAM Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

A. Johnston
Avaya
J. Uberti
Google
J. Yoakum
K. Singh
Avaya

October 27, 2014

# An Origin Attribute for the STUN Protocol
## draft-ietf-tram-stun-origin-02

## Abstract

STUN, or Session Traversal Utilities for NAT, is a protocol used to assist other protocols traverse Network Address Translators or NATs. STUN, and STUN extensions such as TURN, or Traversal Using Relays around NAT, and ICE, Interactive Communications Establishment, have been around for many years but with WebRTC, Web Real-Time Communications, STUN and related extensions are about to see major deployments and implementation due to these protocols being implemented in browsers. This specification defines an ORIGIN attribute for STUN that can be used in similar ways to the HTTP header field of the same name. WebRTC browsers utilizing STUN and TURN would include this attribute which would provide servers with additional information about the STUN and TURN requests they receive. This specification defines the usage of the STUN ORIGIN attribute for web and SIP contexts.

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

## Copyright Notice

Table of Contents

## 1. Introduction

STUN, or Session Traversal Utilities for NAT, is a protocol used to assist other protocols traverse Network Address Translators or NATs. TURN, or Traversal Using Relays around NAT [RFC5766], is a STUN extension [RFC5389] that allows endpoints to acquire a relayed address for media flows. It is most commonly used in conjunction with ICE, Interactive Connectivity Establishment [RFC5245], which is used to establish peer-to-peer flows between endpoints through NATs and firewalls.

STUN defines three authentication modes, depending on the STUN usage. For STUN binding requests sent between peers, such as for ICE connectivity checks, a short term authentication method is recommended. Each peer contributes random strings which are exchanged over signaling and used to authenticate the connectivity checks. For TURN, a usage of STUN used to acquire and refresh relay addresses, a long term authentication method is recommended. This authentication is similar to SIP Digest [RFC3261], which involves an authentication challenge for each request. A server, upon receipt of a TURN request, generates an authentication challenge that includes a realm and nonce. The client resends the TURN request supplying a user name and password based on the realm indicated by the server. For a STUN binding request sent to a STUN server, no authentication is recommended, as generating the response is less work for a server than the server utilizing the short term or long term authentication approach. In addition, the resource requirements of operating a STUN server are minimal.

WebRTC, Web Real-Time Communications, adds peer-to-peer real-time, interactive voice and video media capabilities and data channels to browsers [I-D.ietf-rtcweb-overview] without a plugin or download, and allows web developers to access this functionality using JavaScript API calls [WebRTC-API]. WebRTC includes STUN, TURN, and ICE client functionality built into browsers. For a session established between two browsers, if either browser is behind a NAT, a STUN server is necessary. Public STUN servers are currently available and a web application can suggest a particular STUN server be used. In other cases, a TURN server is needed to establish a peer connection. In this case, TURN credentials need to be available to the browser for the long term authentication approach. A TURN server for WebRTC might serve a number of different domains and realms.

From the perspective of the web application provider, providing service for a number of different domains and realms, it is useful to know something about the source of the STUN request when processing the request. For a web application provider STUN or TURN server, the server will have no idea which web pages or sites are sending binding

requests to the service. In conventional applications, the SOFTWARE attribute would provide some identifying information to the service, but that no longer works when the browser is the application. For a web application provider TURN server, the TURN server does not know which realm to include in an authentication challenge.

In the web world, HTTP requests have the concept of origin. The origin of a web page, as defined in [RFC6454], is defined by the URI's scheme, host or IP address, and port portions. The HTTP Origin header field inserted by the web browser carries this information and is useful information for servers that receive HTTP requests generated via JavaScript. For example, Cross Origin Resource Sharing, CORS, allows an HTTP server to serve HTTP requests from multiple origins.

This specification proposes extending the origin concept to STUN requests. STUN requests generated by a web browser would include the origin of the HTTP page that is initiating the Peer Connection. Using this extra information, a STUN server could use the origin to determine which STUN binding requests to respond to, reducing the load on a STUN server. Using this information, a TURN server could use the origin to determine which realm to include in the authentication challenge. A TURN server can also use the origin information for logging and analytics, and also as additional information after authentication for providing service. This specification also defines an origin for SIP and XMPP users of STUN and TURN.

An important use case that the STUN Origin helps solve is the operation of a multi-tenanted TURN server (i.e. a TURN server that serves multiple, perhaps tens of thousands of different domains). The problem associated with this use case is described in Section 4.5 of [I-D.ietf-tram-auth-problems]. While it is possible for a TURN server to use the same authentication credentials across many domains, a more likely (and more manageable) scenario is to have separate credentials for each domain, and hence a different realm for each domain. With the TURN server configured with a mapping between a domain (conveyed in the Origin) and the realm string (to be used in the authentication challenge), a single TURN URI could be used across all domains, and the resulting JavaScript code would be portable.

Note that the origin information is most useful as a hint in initial STUN and TURN requests as received by a server. However, origin information still has value throughout the session even after authentication for logging and other purposes.

The following sections of this document define the STUN ORIGIN attribute and define its usage.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. STUN ORIGIN attribute

This specification defines how to apply the web origin concept and syntax of [RFC6454] to the STUN protocol.

This specification defines a new Attribute to the STUN protocol [RFC5389]. The attribute is called ORIGIN and uses the syntax defined in Section 15 of [RFC5389]. The number used for this in the type field is 0x802F, chosen in the comprehension optional range. The value of ORIGIN is a variable-length value. It MUST contain a UTF-8 [RFC3629] encoded sequence of characters. Senders MAY include multiple ORIGIN attributes in a request, and receivers MUST support parsing and receiving multiple ORIGIN attributes. The size of ORIGINs included in a STUN message can have a major impact on the size of a STUN packet, and could potentially cause UDP fragmentation. HTTP origins are less than 267 characters (the maximum 253 character domain name plus 8 characters for the URI scheme plus 5 characters for the port number).

For a web browser (HTTP User Agent), the contents of the ORIGIN attribute is the unicode-serialization of an origin defined in Section 6.1 of [RFC6454]. The origin value included is the same as the Origin header field for an HTTP request generated from the web page that is creating the Peer Connection. It does not include any string terminating (\x00) character in the serialization.

For a SIP User Agent [RFC3261] using STUN and TURN, the ORIGIN attribute is set to be the URI of the registrar server used by the User Agent (i.e. the Request-URI of a REGISTER method). This is the full Request-URI component of the SIP ABNF defined in [RFC3261]. For example, a SIP user "sip:bob@biloxi.example.com" might register with the Request-URI of "sip:registrar.biloxi.example.com".

For an XMPP client [RFC6120] using STUN and TURN, the ORIGIN attribute is an XMPP URI [RFC5122] representing the full domainpart of the client’s Jabber ID (JID) as defined in the ABNF of [RFC6122]; for example, if the client’s JID is "juliet@im.example.com/balcony" then the ORIGIN attribute would be "xmpp:im.example.com".

Other contexts can define a usage of the ORIGIN attribute to use an appropriate URI or URL.

If an ORIGIN attribute is not present in a request, it is up to the server how to handle the request. For example, it could assume a default Origin.

### 2.1. STUN Usage

For STUN requests sent without authentication to a STUN server (i.e. STUN binding requests sent to a STUN server), the STUN client SHOULD include the ORIGIN attribute. A STUN server can derive additional information for logging and analytics about the request through the ORIGIN attribute, such as the source of the request. For example, an enterprise STUN server might only reply to STUN binding requests from certain domains.

### 2.2. TURN Usage

For STUN requests sent using the long-term authentication method, such as TURN [RFC5766] allocate requests, the STUN client SHOULD include the ORIGIN attribute. Including the ORIGIN attribute in the Send method is NOT RECOMMENDED. A TURN server can use the ORIGIN attribute to determine which REALM to include in the authentication challenge. A TURN server can also use the ORIGIN attribute after authentication to provide appropriate service. See the section below on "Multiple Origins."

### 2.3. NAT Behavior Discovery Usage

For the NAT Behavior Discovery Usage in [RFC5780], the ORIGIN attribute SHOULD be included in requests sent to a STUN server. This usage is most similar to the STUN Usage described earlier.

### 2.4. ICE Usage

For STUN requests sent using the short-term authentication method, such as ICE connectivity checks [RFC5245], the use of the ORIGIN attribute is NOT RECOMMENDED. No valid use cases for the ORIGIN attribute have been identified to date.

### 2.5. Media Keep-Alive Usage

For media keep-alive STUN requests described in Section 20 of [RFC5245], the use of the ORIGIN attribute is NOT RECOMMENDED. No valid use cases for the ORIGIN attribute have been identified to date.

### 2.6. SIP Keep-Alive Usage

For SIP keep-alive STUN requests described in [RFC5626], the use of the ORIGIN attribute is NOT RECOMMENDED. No valid use cases for the ORIGIN attribute have been identified to date.

### 2.7. Multiple Origins

Multiple Origins for HTTP Requests are described in Section 7.2 of [RFC6454]. Multiple origins can occur when the same resource is fetched by multiple origins at the same time (e.g. multiple tabs, windows, frames, etc.). In the context of WebRTC, it doesn't make sense for a STUN binding or TURN allocation to be shared across origins (e.g. Peer Connections). Based on their definitions, multiple SIP and XMPP origins also do not apply here.

## 3. IANA Considerations

This specification, if approved, adds a new value to the IANA "STUN Attributes Registry" created by [RFC5389]. The ORIGIN attribute value is 0x802F.

## 4. Security Considerations

The security considerations of [RFC6454] apply to this extension. Servers using the information present in the STUN ORIGIN attribute need to realize that this attribute could be set arbitrarily by a non-browser client or modified by an intermediary. The method proposed in this document is not meant to replace existing STUN authentication mechanisms but to provide additional information to the server for logging and analytics and how to handle the request after authentication.

Just as browsers do not allow a web application to set the Origin header field via JavaScript, browsers should not allow a web application through JavaScript to set the STUN ORIGIN attribute.

While the STUN MESSAGE-INTEGRITY attribute can provide integrity protection for all attributes present in a STUN request, MESSAGE-INTEGRITY is not present in the initial STUN message sent. As a result, an ORIGIN attribute could be modified or removed from a STUN request without the server knowing. DTLS or TLS transport SHOULD be used when integrity protection for the ORIGIN attribute is important.

The STUN ORIGIN attribute does have privacy implications. The recipient of the STUN request learns the web origin of the user. In

addition, an on-path attacker could determine this information by inspecting STUN messages between the STUN client and STUN server, depending on the transport used. This information is often available in other messages sent by the browser, such as DNS or HTTP requests. However, in cases where secure HTTP is used, including the ORIGIN attribute over an unencrypted transport could leak this information. STUN has a defined TLS transport; however, TLS transport is generally unsuitable for the real-time media flows that follow STUN requests and must use the same transport. The DTLS transport for STUN [I-D.ietf-tram-stun-dtls] provides a very good privacy solution to this problem. In cases where privacy is paramount, the ORIGIN attribute SHOULD NOT be included or only included if DTLS or TLS transport is used.

The STUN ORIGIN attribute also has privacy implications in that the origin information is shared with a STUN or TURN server which otherwise might not know this information. This information could be used to track usage of real-time communication services. A STUN or TURN server will always know the public IP address of each user, but the ORIGIN attribute provides more information about which service or provider is being used. The particular STUN and TURN servers used are selected by the real-time communications service provider (i.e. the web provider for WebRTC or the SIP or XMPP service provider). In addition, they are usually also run by the same provider, or by a trusted partner, especially for TURN. However, a service or provider using a public STUN server needs to recognize that the operator of the public STUN server will learn the identity of the service or provider through this extension.

## 5. Implementation Status

Note to RFC Editor: Please remove this entire section prior to publication, including the reference to RFC 6982.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Two proof-of-concept implementations have been created in support of this proposed standard. One provides a WebRTC enabled browser that includes the appropriate STUN ORIGIN Attribute with the Origin insight known to the browser in STUN/TURN messages sent to servers. The other provides an example of a multiple realms capable TURN server that takes advantage of Origin insight provided in the STUN ORIGIN Attribute.

A Chrome browser implementation has been created by Graham Yoakum and Ryan Yoakum (Skobalt LLC) and is freely licensed under the standard terms of the open source Chromium and WebRTC projects. This proof-of-concept version of the Google Chrome browser (nicknamed 'Chromeo') sends Origin insight in STUN and TURN messages using the proposed new STUN ORIGIN attribute with a value of 0x802F (as initially proposed, however that value is easily changed in a single line of code). 'Chromeo' includes a Chrome flag to enable and disable this unique feature (and is by default disabled to prevent any non-intentional use of this feature until the standard is finalized). This implementation is based on is draft-johnston-tram-stun-origin-02.

Coordinated changes to both the WebRTC and Chromium open source projects have been formally submitted for consideration. The two submitted change lists together implement the complete browser proof-of-concept. 'Chromeo' has been built for Linux and STUN protocol behavior has been verified using WireShark traces illustrating that proper STUN Origin attributes are being included in STUN/TURN messages sent by the browser to servers (screen captures of STUN messages illustrating the Origin attribute and content are available).

The WebRTC and Chromium open source projects can be found at: http://www.webrtc.org/ and http://www.chromium.org/

Google can choose to accept or modify the changes proposed for Chrome and other browser vendors can access and take advantage of the publicly available WebRTC and Chromium open source submissions as desired. Hopefully this will enable browsers to quickly implement STUN Origin enhancements.

A multiple realms capable advanced open source Origin enabled TURN server (named 'Coturn') has been created by Oleg Moskalenko and is

freely licensed under the New BSD license. This reference implementation and proof-of-concept provides a clone (a spin-off) of the rfc5766-turn-server project adding Origin-based multiple realms support.

'Coturn' is backward-compatible with rfc5766-turn-server project but the code is more complex and it uses a different (also more complex) database structure. It is the intent to add all IETF TRAM TURN server related capabilities to this project as they mature. 'Coturn' is publicly available and can be found at: https://code.google.com/p/coturn/

## 6. Acknowledgements

Thanks to John Selbie, Tirumaleswar Reddy, Simon Perreault, Marc Petit-Huguenin, Andy Hutton, and Oleg Moskalenko for their feedback and reviews. Special thanks to Graham Yoakum and Ryan Yoakum of Skobalt LLC and Oleg Moskalenko of rfc5766-turn-server project for contributing open source proof-of-concept implementations for a Chrome web browser and a multiple realms capable TURN server, quickly demonstrating feasibility.

## 7. Normative References

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-12 (work in progress), October 2014.

[I-D.ietf-tram-auth-problems]
Reddy, T., R, R., Perumal, M., and A. Yegin, "Problems with STUN long-term Authentication for TURN", draft-ietf-tram-auth-problems-05 (work in progress), August 2014.

[I-D.ietf-tram-stun-dtls]
Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)", draft-ietf-tram-stun-dtls-05 (work in progress), June 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] 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.

[RFC3489] 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.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4020] Kompella, K. and A. Zinin, "Early IANA Allocation of Standards Track Code Points", RFC 4020, February 2005.

[RFC5122] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)", RFC 5122, February 2008.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, May 2010.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, March 2011.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[WebRTC-API] Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", W3C Working Draft http://www.w3.org/TR/webrtc/, 2013, <http://www.w3.org/TR/2013/WD-webrtc-20130910/>.

## Authors' Addresses

Alan Johnston
Avaya
St. Louis, MO
USA

Phone:
Email: alan.b.johnston@gmail.com

Justin Uberti
Google
Kirkland, WA
USA

Phone:
Email: justin@uberti.name

John Yoakum
Avaya
Cary, NC
USA

Phone:
Email: yoakum@avaya.com

Kundan Singh
Avaya
San Francisco, CA
USA

Phone:
Email: kundan10@gmail.com

TRAM
Internet-Draft
Intended status: Standards Track
Expires: March 19, 2015

P. Martinsen
Cisco
J. Uberti
Google
O. Moskalenko
Unaffiliated

September 15, 2014

# Single SOcket Dual Allocation with TURN
draft-martinsen-tram-ssoda-01

## Abstract

This draft describes a simple method for allocating one IPv4 and one IPv6 relay address from a single ALLOCATE request to the TURN server. This saves local ports on the client, reduces the number of candidates gathered by the client, and reduces the number of messages sent between the client and the TURN server.

## Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 19, 2015.

## Copyright Notice

Table of Contents



## 1. Introduction

The main motivation for this draft is to reduce the number of local ports on the client, reduce the number of candidates gathered during the discovery process, and reduce the number of messages that need to be exchanged to allocate the relay addresses needed for ICE.

Reducing the number of local ports is important as it saves resources at three places in the network. First, the number of open ports on the client is reduced, leading to fewer host candidates. Secondly, with fewer local host ports there will be fewer NAT bindings for the NAT to keep track of, and fewer server reflexive candidates. Lastly, with a single 5-tuple in use, it reduces the number of open ports the TURN server needs to open on the interface towards the client (Private side). As ports are a scarce resource (16-bit number)

preserving them on the NAT and a the TURN server can make large scale deployments easier.

## 2. Creating an Allocation

The behavior specified here affects the processing defined in Section 6 of [RFC5766] and Section 4 of [RFC6156].

### 2.1. Sending an Allocate Request

A client that wishes to obtain one IPv6 and one IPv4 by sending one Allocate request MUST include two REQUESTED-ADDRESS-FAMILY attributes, one for each address family, in the Allocate request that it sends to the TURN server. The order of the REQUESTED-ADDRESS-FAMILY is arbitrary, because the server either understands SSODA (then the order does not matter) or the server does not understand SSODA (then the server behavior is undefined - it may return a 400 error, or it may take the first attribute, or it may take the last attribute). Multiple candidates of the same family are not supported; the client MUST NOT include more than one REQUESTED-ADDRESS-FAMILY attribute for a given address family. The mechanism to formulate an Allocate request is described in Section 6.1 of [RFC5766].

The SSODA mechanism is not available when using the odd/ even port allocation scheme. Clients MUST NOT include a REQUESTED-ADDRESS-FAMILY attribute in an Allocate request that contains a RESERVATION-TOKEN attribute. Clients MUST NOT include a second REQUESTED-ADDRESS-FAMILY attribute in an Allocate request that contains an EVEN-PORT attribute.

### 2.2. Receiving an Allocate Request

Once a server has verified that the request is authenticated and has not been tampered with, the TURN server processes the Allocate request following the rules in [RFC5766] and [RFC6156].. Only one REQUESTED-ADDRESS-FAMILY attribute with the same family value is allowed in the request. If two attributes with the same family value exist the server MUST return 400 Bad Request error.

If no REQUESTED-ADDRESS-FAMILY attributes are present, the server MUST treat this as if the request contained a single REQUESTED-ADDRESS-FAMILY specifying the IPv4 address family.

If the server can successfully process the request, it allocates a relay address for each of the REQUESTED-ADDRESS-FAMILY attributes present in the Allocate request. The allocated relay addresses are returned in separate XOR-RELAYED-ADDRESS attributes in the Allocate

response message. The ordering of the XOR-RELAYED-ADDRESS attributes in the response is arbitrary.

If the server cannot satisfy the request at all, because none of the specified address families are supported, the server MUST return a 440 error code, as indicated in [RFC6156].

If the server cannot satisfy the request at all, because the server could not allocate any of the specified addresses, the server MUST return a 508 (Insufficient Capacity) error code as indicated in [RFC5766].

If some of the requested address could be allocated, but some could not, either because the requested address family is not supported, or the server currently lacks capacity, the server MUST indicate this partial success by returning an Allocate Success Response that contains XOR-RELAYED-ADDRESS attributes for the addresses that were successfully allocated, as well as XOR-RELAYED-ADDRESS with ANY addresses (that is, IPv4 address 0.0.0.0:0 or IPv6 address [::0]:0) corresponding to the address families that could not be allocated. This will notify the client that the desired REQUESTED-ADDRESS-FAMILY was understood, but could not be allocated. A success response with ANY addresses MUST NOT be returned if all allocation requests cannot be satisfied; instead, an error response should be returned, as indicated above.

This somewhat unusual pattern of partial success is used to avoid the need for an additional round-trip when the client just wants whatever address families the TURN server supports.

Note that while allocating multiple address families at the same time is supported, doing this sequentially is not. The server MUST reject any attempt to "add" an address family to an existing allocation with a 437 (Allocation Mismatch) error code.

[OPEN ISSUE 1: do we need to include REQUESTED-ADDRESS-FAMILY attribute(s) with failed address family (or families) to help the client to recognize whether this is an "old" non-SSODA server or a "new" SSODA-supporting server ?]

[OPEN ISSUE 2: do we have to consider a particular ordering of REQUESTED-ADDRESS-FAMILY and REQUESTED-ADDRESS-FAMILY attributes in the ALLOCATE request and response ? Can attribute ordering provide some benefits in this case ?]

### 2.3. Receiving an Allocate Success Response

This section describes how the client must react on receiving a response to the dual allocation request. If the client is not using dual allocation, then the behavior is the same as the rules in [RFC5766] and in [RFC6156].

If the client receives an Allocate Success Response containing a non-ANY (ANY as defined above) XOR-RELAYED-ADDRESS attribute for each of the REQUESTED-ADDRESS-FAMILY attributes in the Allocate request sent by the client, the client knows that the TURN server supports multiple address family allocation over a single socket. All relay addresses can now be used by the client.

If the Allocate response contains both usable XOR-RELAYED-ADDRESS attributes as well as ANY XOR-RELAYED-ADDRESS attributes, then the client knows that the TURN server "understands" dual allocation SSODA request, but the server either does not support one of the requested address families or cannot currently allocate an address of that family. The allocated non-ANY address can be used, but the client SHOULD NOT try to allocate any of the unsupported families on a different 5-tuple.

If the Allocate Response contains only one XOR-RELAYED-ADDRESS attribute, then the client knows that the TURN server does not support SSODA. The client can retry the missing address family allocations on new 5-tuples, if desired. Subsequent Allocate requests towards the same TURN server SHOULD NOT include multiple REQUESTED-ADDRESS-FAMILY attributes.

### 2.4. Receiving an Allocate Error Response

When doing dual allocation, if the client receives an Allocate error response with the 440 (Unsupported Address Family) error code, then the client knows that the TURN server does not support any of the desired address families, or might be a non-SSODA server that misinterpreted the included REQUESTED-ADDRESS-FAMILY attributes in the Allocate request. The client SHOULD retry its IPv4 request on the same 5-tuple, with no REQUESTED-ADDRESS-FAMILY attribute, and MAY retry other address families on different local ports, by sending an Allocate request with only one REQUESTED-ADDRESS-FAMILY attribute.

## 3. Refreshing an Allocation

The behavior specified here affects the processing defined in Section 7 of [RFC5766] and Section 5 of [RFC6156]. This section MUST only be used if the client has verified that the TURN server supports

SSODA during the allocation creation described in Section 2.1. Otherwise, revert back to RFC 5766 or RFC 6156 behavior.

### 3.1. Sending a Refresh Request

To perform an allocation refresh, the client generates a Refresh Request as described in Section 7.1 of [RFC5766]. When refreshing a dual allocation, the client SHOULD include one or more REQUESTED-ADDRESS-FAMILY attributes describing the the family types that should be refreshed; the client MUST only include family types that it previously allocated and has not yet deleted. When refreshing a (single) allocation on a server that does not not support SSODA, REQUESTED-ADDRESS-FAMILY should be omitted, for backwards compatibility.

This process can also be used to delete an allocation of a specific address type, by setting the lifetime of that refresh request to 0. It is possible to delete one or more allocations depending on how many REQUESTED-ADDRESS-FAMILY attributes are included. Deleting a single allocation destroys any permissions or channels associated with that particular allocation; it MUST NOT affect any permissions or channels associated with allocations for other address families.

### 3.2. Receiving a Refresh Request

The server refreshes the allocated address families that match the supplied REQUESTED-ADDRESS-FAMILY values. If any of the values in the request do not match a currently allocated address, the server MUST respond with a 437 (Allocation Mismatch) error. [OPEN ISSUE: discuss whether this is the right error code for the situation] If no REQUESTED-ADDRESS-FAMILY is present, the request should be treated as applying to all current allocations, for backward compatibility.

The server MUST then refresh or delete the specified allocations, and return a Refresh Success Response.

### 3.3. CreatePermission

The behavior specified here affects the processing defined in Section 9 of [RFC5766] and Section 6 of [RFC6156]

#### 3.3.1. Sending a CreatePermission Request

The client MUST only include XOR-PEER-ADDRESS attributes with addresses that match an address family of one of the currently allocated addresses.

#### 3.3.2. Receiving a CreatePermission Request

If an XOR-PEER-ADDRESS attribute contains an address of an address family different than that any of the relayed transport addresses allocated, the server MUST generate an error response with the 443 (Peer Address Family Mismatch) response code, which is defined in Section 6.2.1 of [RFC6156].

## 4. Channels

The session channels setup process follows the same rules as in [RFC5766] and in [RFC6156]; the client is allowed to set up multiple channels within the same 5-tuple session. However, when using SSODA and dual allocation, the peer addresses of those channels may be of different families. Thus, a single 5-tuple session may create several IPv4 channels and several IPv6 channels.

## 5. Implementation Status

[Note to RFC Editor: Please remove this section and reference to [RFC6982] prior to publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

### 5.1. open-sys

Organization: This is a public project, the full list of authors and contributors here: http://turnserver.open-sys.org/downloads/AUTHORS

Description: A mature open-source TURN server specs implementation (RFC 5766, RFC 6062, RFC 6156, etc) designed for high-performance applications, especially geared for WebRTC.

Implementation: http://code.google.com/p/coturn/

Level of maturity: The TURN SSODA extension implementation can be qualified as "production" - it is well tested and fully implemented, but not widely used, yet..

Coverage: Fully implements SSODA this draft.

Licensing: BSD: http://turnserver.open-sys.org/downloads/LICENSE

Implementation experience: Few changes to existing code

Contact: Oleg Moskalenko <mom040267@gmail.com>.

### 5.2. NATTools

Organization: Cisco

Description: NATTools is a set of client side focused ICE/TURN/STUN libraries.

Implementation: https://github.com/cisco/NATTools

Level of maturity: Running test code works well. Not tested in any released products.

Coverage: Implement this draft.

Licensing: BSD

Implementation experience: Simple, few changes to the client.

Contact: Paal-Erik Martinsen <palmarti@gmail.com>.

## 6. Security Considerations

As the client can ask for two allocations for each allocation request sent, the TURN server can be DOS attacked with fewer messages. However this problem is minimal as the messages needs to be authenticated first as described in RFC 5766 [RFC5766].

## 7. Acknowledgements

Authors would like to thank Simon Perreault for providing ideas direction and insight. Jonathan Lennox provided excellent feedback on the mailing list.

## 8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6156] Camarillo, G., Novo, O., and S. Perreault, "Traversal Using Relays around NAT (TURN) Extension for IPv6", RFC 6156, April 2011.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

## Authors' Addresses

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens vei 20
Lysaker, Akershus 1366
Norway

Email: palmarti@cisco.com

Justin Uberti
Google
Kirkland, WA
USA

Email: justin@uberti.name

Oleg Moskalenko
Unaffiliated
Walnut Creek, CA
USA

Email: mom040267@gmail.com
URI: https://code.google.com/p/coturn/

TRAM
Internet-Draft
Intended status: Informational
Expires: April 29, 2015

P. Patil
T. Reddy
G. Salgueiro
Cisco

October 26, 2014

# Traversal Using Relays around NAT (TURN) Server Selection draft-patil-tram-turn-serv-selection-00

## Abstract

A TURN client may discover multiple TURN servers. In such a case, there are no guidelines that a client can follow to choose or prefer a particular TURN server among those discovered. This document details selection criteria, as guidelines, that can be used by a client to perform an informed TURN server selection decision.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 29, 2015.

## Copyright Notice

Table of Contents



## 1. Introduction

Using any of the discovery mechanisms described in [I-D.ietf-tram-turn-server-discovery], a client may discover multiple Traversal Using Relays around NAT (TURN) servers. The TURN servers discovered could be provided by an enterprise network, an access network, an application service provider or a third party provider. Therefore, the client needs to be able to choose a TURN server that best suits its needs.

Selection criteria could be based on parameters such as:

- Security
- Location Privacy
- Authentication
- User Experience
- Interface Selection (if the client is multi-interfaced)
- Mobility Support

This document describes procedures that a client can use to choose the most appropriate TURN server based on any one or more

combinations of the above parameters. A client could also use the aforementioned selection criteria to prioritize the discovered TURN servers based on these parameters if backup servers are implemented for added resiliency and robustness.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 3. TURN Server Selection Criteria

The accessibility of possible TURN servers SHOULD be tested and verified prior to beginning Interactive Connectivity Establishment (ICE) [RFC5245]. Any TURN servers that fail such accessibility tests (including credentials verification) SHOULD be excluded. These early tests are an often an optimal opportunity to calculate performance metrics, such as the round-trip time (RTT), that might be used as TURN server prioritization factors, as discussed in Section 3.3. Throughout the lifetime of the application, it is RECOMMENDED to periodically test the entire selection list, in case better TURN servers suddenly appear or connectivity to others is unexpectedly lost.

The parameters described in this Section are intended as TURN server selection criteria or as weighting factors for TURN server prioritization.

### 3.1. Local Configuration

Local or manual configuration takes precedence for TURN server selection. A client could be configured with an explicit preferred list of TURN servers. Local configuration could list servers in order of preference. For example, a TURN client could opt for a TURN server offered by the Enterprise and fall back to a TURN server offered by the Internet Service Provider (ISP) or a cloud service if the Enterprise TURN server wasn’t available.

An implementation MAY give the user an opportunity (e.g., by means of configuration file options or menu items) to specify this preference.

### 3.2. Security

If a TURN client wants security for its connections, it should opt for a TURN server that supports the usage of Transport Layer Security (TLS) [RFC5246] and Datagram Transport Layer Security (DTLS) [RFC6347] as a transport protocol for Session Traversal Utilities for

NAT (STUN), as defined in [RFC5389] and [RFC7350]. If multiple servers offer this support, the client could use Location Privacy (Section 3.2.1) and Authentication (Section 3.2.2) criteria to determine which among the list is most suitable.

The need for security depends on the type of connected network (i.e., whether the host is connected to a home network versus an Enterprise network versus a coffee shop network). It is recommended that a client always choose security, but this condition could vary depending on the degree of trust with the connected network.

### 3.2.1. Location Privacy

In addition to security, a TURN client may require additional location privacy from an external peer.

Scenario 1: A client may not wish to use a TURN server in its Enterprise or access network because the client location could be determined by the external peer. In such a case, the client may choose to use a distributed multi-tenant or a cloud-based TURN server that can provide privacy by obscuring the network from which the client is communicating with the remote peer.

Scenario 2: A TURN client that desires to perform Scenario 1, but cannot because of firewall policy that forces the client to pick Enterprise-provided TURN server for external communication, can use TURN-in-TURN through the enterprise’s TURN server as described in [I-D.schwartz-rtcweb-return].

Location privacy may not be critical if the client attempts to communicate with a peer within the same domain.

### 3.2.2. Authentication

A TURN client should prefer a TURN server whose authenticity can be ascertained. A simple certificate trust chain validation during the process of (D)TLS handshake should be able to validate the server.

A TURN client could also be pre-configured with the names of trusted TURN servers. When connecting to a TURN server, a TURN client should start with verifying that the TURN server name matches the pre-configured list of TURN servers, and finally validating its certificate trust chain. For TURN servers that don’t have a certificate trust chain, the configured list of TURN servers can contain the certificate fingerprint of the TURN server (i.e., a simple whitelist of name and certificate fingerprint).

DNS-based Authentication of Named Entities (DANE) can also be used to validate the certificate presented by TURN server as described in [I-D.petithuguenin-tram-stun-dane].

### 3.3. User Experience

All else being equal (or if a TURN client is able to converge on a set of TURN servers based on parameters described in Section 3.2), a TURN client should choose a TURN server that provides the best user experience at that point in time (based on factors such as RTT, real-time clock (RTC), etc).

If using ICE regular nomination, ICE connectivity check round-trip time can influence the selection amongst the valid pairs. This way a candidate pair with relayed candidate could be selected even if it has lower-priority than other valid pairs.

### 3.4. Interface

With a multi-interfaced node, selection of the correct interface and source address is often crucial. How to select an interface and IP address family is out of scope for this document. A client could account for the provisioning domain described in [I-D.ietf-mif-mpvd-arch] to determine which interface to choose.

### 3.5. Mobility Support

If a TURN client is aware that the host is mobile, and all other parameters being equal, the client SHOULD choose a TURN server that supports mobility [I-D.wing-tram-turn-mobility].

## 4. Security Considerations

This document does not itself introduce security issues, rather it merely presents best practices for TURN server selection. Security considerations described in [RFC5766] are applicable to for all TURN usage.

## 5. Acknowledgements

The authors would like to thank Dan Wing, Marc Petit-Huguenin for their review and valuable comments.

## 6. References

## 6.1. Normative References

[I-D.ietf-mif-mpvd-arch]
Anipko, D., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-07 (work in progress), October 2014.

[I-D.ietf-tram-turn-server-discovery]
Patil, P., Reddy, T., and D. Wing, "TURN Server Auto Discovery", draft-ietf-tram-turn-server-discovery-00 (work in progress), July 2014.

[I-D.petithuguenin-tram-stun-dane]
Petit-Huguenin, M. and G. Salgueiro, "Using DNS-based Authentication of Named Entities (DANE) to validate TLS certificates for the Session Traversal Utilities for NAT (STUN) protocol", draft-petithuguenin-tram-stun-dane-02 (work in progress), October 2014.

[I-D.schwartz-rtcweb-return]
Schwartz, B., "Recursively Encapsulated TURN (RETURN) for Connectivity and Privacy in WebRTC", draft-schwartz-rtcweb-return-03 (work in progress), September 2014.

[I-D.wing-tram-turn-mobility]
Wing, D., Patil, P., Reddy, T., and P. Martinsen, "Mobility with TURN", draft-wing-tram-turn-mobility-02 (work in progress), September 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC7350] Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)", RFC 7350, August 2014.

## 6.2. Informative References

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

## Authors' Addresses

Prashanth Patil
Cisco Systems, Inc.
Bangalore
India

Email: praspati@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Gonzalo Salgueiro
Cisco
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com

tram
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

A. Wang
China Telecom
B. Liu
Huawei Technologies

October 27, 2014

# A Lightweight TURN Architecture and Specification (TURNLite) draft-wang-tram-turnlite-01

## Abstract

This document proposes a lightweight TURN architecture which simplifies the application provider side complexity of implementing TURN server by transfering the data relay processing to the ISP infrastructure (e.g. CGN). To achieve this goal, a new "Couple" operation using STUN message format is also defined.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

## Copyright Notice

Table of Contents



## 1. Introduction

With the depletion of public IPv4 addresses and the deployment of more and more NAT devices in real network, the communication between two hosts that located behind NAT devices is becoming more difficult. IPv6 adoption and the coexistence of IPv4 and IPv6 hosts within one network exacerbate this issue. TURN [RFC5766] technology is aim to solve these problems, but currently its deployment is very limited and most of application provider user their own platform to transfer the data between two hosts that behind NAT environment and to translate the communication packets between two hosts in different address family.

The data relay device deployed centrally by various application providers often lead to the inefficient data transmit between two hosts. The relay device must deal with complex network layered problems with which the application providers are not familiar.

On the other hand, service provider deploys many CGN devices in a distributed manner within their networks. If the service provider can use these CGN devices as the relay devices for communication between two hosts behind NATs or that from different address families, and open their data translation/forwarding capability to the application providers, the host to host communication will be easier and more efficient, and the deployment of IPv6 technology will also be accelerated.

This document proposes a lightweight TURN architecture which simplifies the application provider side complexity of implementing TURN server by transfering the data relay processing to the ISP infrastructure (mainly regarding to CGN (Carried Grade NAT) ). To achieve this goal, a new "Couple" operation using STUN message format is defined. The "Couple" operation could make the CGNs be able to couple two separate TCP/UDP connections that from the host to CGN device and relay data between two hosts. The "couple" operation could be considered as a common interface that invoked by the application providers to exploit the capabilities of CGN devices that deployed in a distributed manner by service provider.

## 2. Requirements Language and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

- Application Provider: the service providers who provide client to client communications through the Internet. E.g. VoIP service providers, instant message service providers etc.

- TURNLite: lightweight TURN architecture. The word "lightweight" is in the perspective of an application provider.

- TURNLite Client: the TURNLite entity that deployed in the application providers' networks; be responsible for TURNLite signaling/control interactions with the TURNLite servers.

- TURNLite Server: the TURNLite entity that deployed in the ISP's networks; be mainly responsible for the data relay between an application providers' clients. Normally, the TURNLite servers collated with the CGNs (Carrier Grade NATs) within the service provider.

## 3. Targeted Problems

### 3.1. Application Providers Need to Deploy TURN servers individually

In real practice, TURN servers are deployed and managed by each application provider separately within their own networks. These TURN servers are responsible for all the signaling and data relay function between their clients. Thus, the application providers might face the following challenges:

- The application providers need to be capable of dealing with large amount of relaying traffic, which is a significant burden for them.
- At the early stage of an application/service development, normally the application provider is only able to deploy centralized TURN servers due to budget and capacity limitation. Thus, the traffic path between distributed clients through the central TURN servers might be inefficient.
- TURN servers need to reserve a large amount of relay addresses which are expensive resources now and are difficult for the application providers to apply and manage.

These challenges can be mitigated by utilizing the CGN devices that deployed within the service provider’s network to fulfill the data relay function needed by the application provider. The service provider is expert in the network/transport layer packet processing and has large amounts of resources to use and schedule. They can provide such capability to the application provider and alleviate them from the complicate and laborious work of deploying TURN server.

### 3.2. Dedicated Solutions for Different Requirements

Different client-to-client communication scenarios require dedicated adaption mechanisms respectively in current TURN solution. For example, for the following three communication scenarios, there are specific TURN extensions under developing.

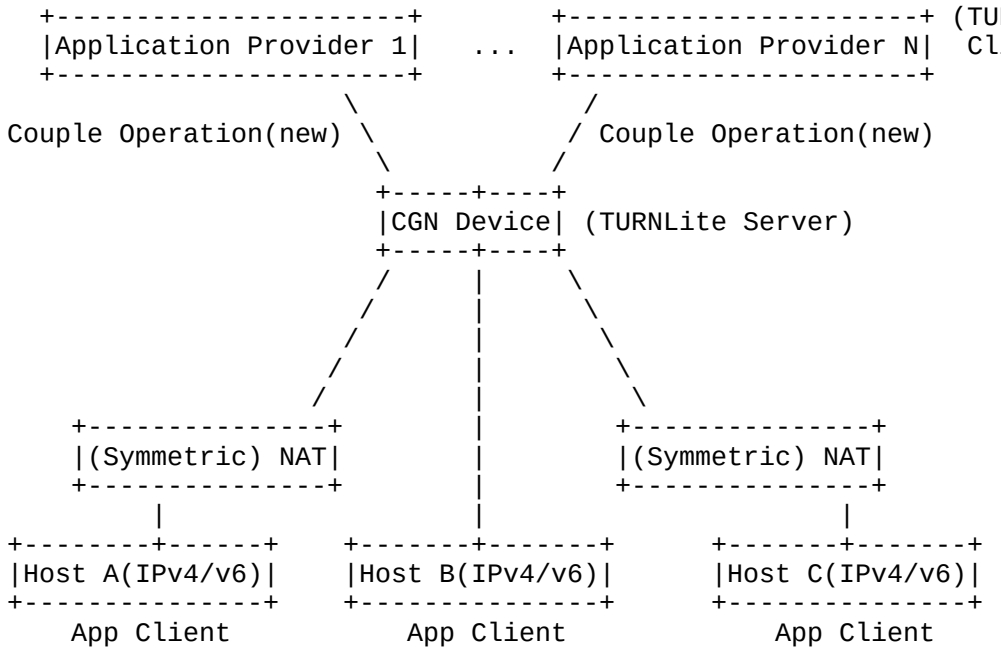
- TCP-based client-to-client communication, see [RFC6062]
- IPv6 client-to-client communication, see [RFC6156].
- Host mobility, see [I-D.wing-tram-turn-mobility].

These dedicated extension mechanisms will obviously increase the complexity of implementing the TURN server. The complexity would become blocking factor of the TURN adoption by real practice.

Thus, there should be one lightweight TURN solution to cope with these challenges and provide one unified solution to fulfill the client-to-client communication under various scenarios.

## 4. Solution Overview

### 4.1. TURNLite Reference Architecture




As the figure shows, the main difference between the TURN and TURNLite is the data relay function of original TURN server is offloaded to the CGN device. the TURNLite Server/CGN device provides such capabilities to the Application Provider via the newly defined Couple operation. The Application Provider does not need to deploy a full TURN server individually, they only need to apply their requirements to the TURNLite Server/CGN device via the common control interface. The subsequent data between the application clients will be relayed by the CGN devices according to the couple information provided in the Couple operation.

Symmetric NAT traverse is supported in TURNLite. IPv6 and IPv4 host communication is also supported.

### 4.2. Solution Rationale

The solution could be briefly described as the following stages:

1) The hosts communicate with the application server respectively. The communication might be based on the application private protocol. At this stage, the key points are:

- The application server learns the reflex address of the hosts.

- The application server learns that Host A wants to communicate with Host C. Thus, it could correlates the reflex addresses of the two hosts.

- The application server selects a proper CGN based on the pair of reflex addresses and indicates the CGN address to host A and C.

2) Host A and C punch a hole in the NAT to the given CGN respectively (this could be a standard STUN [RFC5389] process, for example, sending the STUN binding messages to the given CGN). And report their reflex addresses that bound to the CGN's well-known transport address to the application server.

3) The application server use the newly defined "Couple" operation to couple the two newly acquired reflex addresses bound to the CGN reported by the hosts. Then the hosts could directly send the data to the CGN, and the CGN could relay the data correctly according to the couple information.

## 5. TURNLite Communication Procedures

### 5.1. Procedures of Communication Traversing Symmetric NATs

When one of the communication hosts located behind the symmetric NAT device, the host-to-host communication must via one relay device. Below are the key procedures of TURNLite solution, we use the similiar figure that described in [I-D.ietf-tram-turnbis] for comparison.

Please note that the figure does not include the application server which acts as the TURNLite client. The communication procedures between the hosts and the application server are not included as well, since they might be private communication protocol/mechanism developed by the application provider.

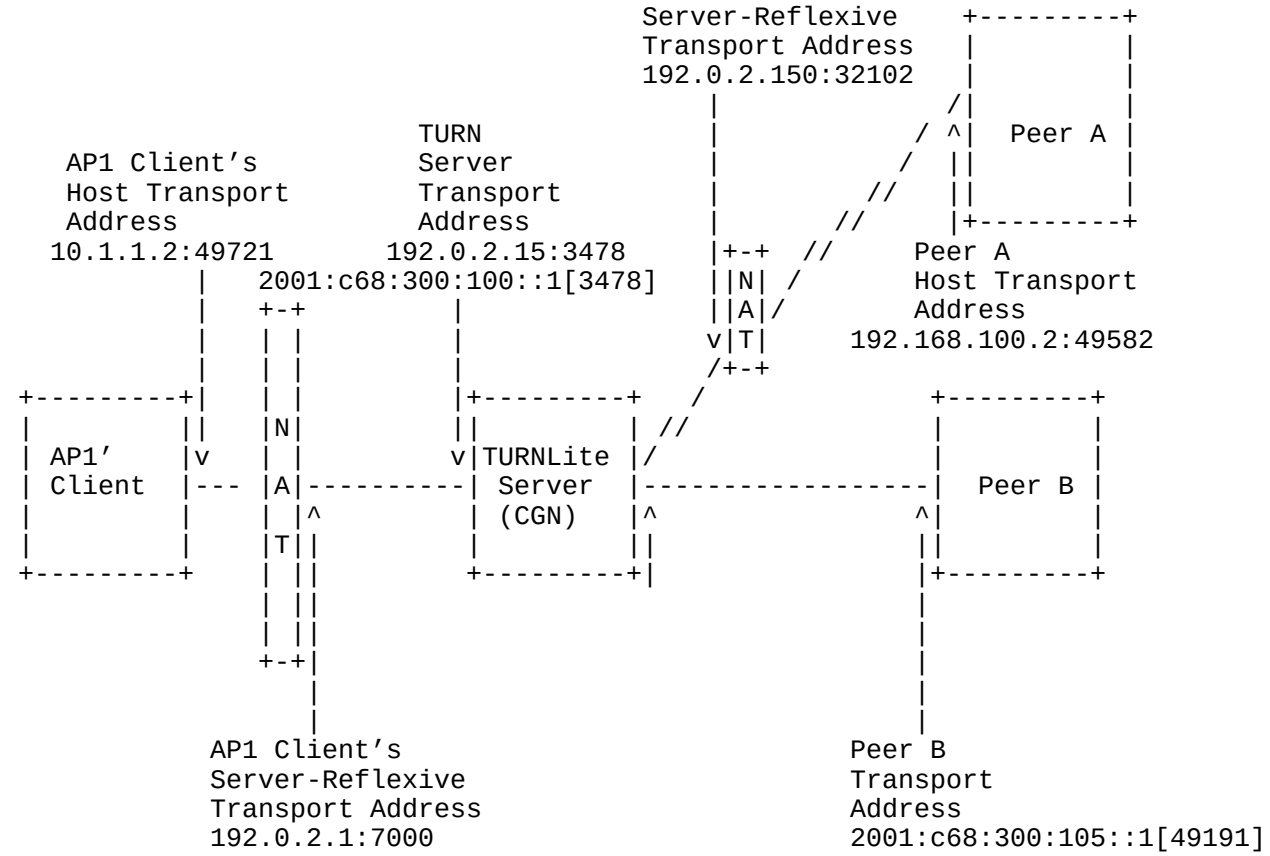



Figure 5-1 Example of TURNLite Communication Scenarios

Communication Procedures:

(Note: before the following step one, the two AP1's clients both have communicated with the application server. The communication might be based on the application private protocol. And the application server needs to select a proper CGN for the clients based on their reflex addresses the server learned. Currently, this document doesn't specifically consider these problems. )

1. If AP1's Client and Peer A want to communicate with each other, they will send STUN binding message to the well-known public IPv4 relay address/port (192.0.2.15:3478 in this example, and it is just the selected CGN's address), get their reflex public addresses to this relay address(that is 192.0.2.1:7000 and 192.0.2.150:32102 respectively).

2. AP1 client and Peer A should report their reflex address to Application Provider 1.

3. Application Provider 1 will use the mapped address of AP1 Client and Peer A to formulate one Couple message that defined in this draft, send it to the CGN device, let the CGN device to build one Couple table for AP1 Client and Peer A. The Couple table looks like below:

| Reflextive transport address of AP1's Client | Reflextive transport address of Peer A | Transport Protocol |
|---|---|---|
| 192.0.2.1:7000 | 192.0.2.150:32102 | TCP/UDP |

Table 5-1: Couple Table Example (Symmetric case)

4. AP1 Client will send the AP1 application data to the well-known public IPv4 relay address/port(192.0.2.15:3478 in this example) of the CGN device.

5. CGN device will look up the Couple table that formulated in the step c), use the source address of received packet (192.0.2.1:7000 in this example) to get the reflex IPv4 address of Peer A. It then will change the source address of the packet to the well-known public IPv4 relay address of CGN device, the destination address of this packet to the IPv4 reflex address of Peer A.

6. The translated packet will be forwarded to the Peer A, processed by the AP1 client located on it.

7. The return traffic will also be sent to the well-known IPv4 relay address/port of CGN device, converted by the CGN device, and sent back to the Application Provider1's Client.

## 5.2. Procedures of IPv4 and IPv6 Host Communication

1. If AP1 clients and Peer B want to communicate with each other, they will send STUN binding message to the well-known public IPv4 relay address/port (192.0.2.15:3478 in this example), get their reflex public addresses to this relay address (that is 192.0.2.1:7000 and 2001:c68:300:105::1[49191]respectively).

2. AP1 client and Peer B should report their reflex address to Application Provider 1.

3. Application Provider 1 will use the mapped address of AP1 Client and Peer B to formulate one Couple message that defined in this draft, send it to the CGN device, let the CGN device to build one Couple table for AP1 Client and Peer B. The Couple table looks like below:

| Reflextive transport address of AP1's Client | Reflextive transport address of Peer B | Transport Protocol |
|---|---|---|
| 192.0.2.1:7000 | 2001:c68:300:105::1[49191] | TCP/UDP |

Table 5-2: Couple Table Example (Case of different address families)

4. AP1 Client will send the AP1 application data to the well-known public IPv4 relay address/port(192.0.2.15:3478 in this example) of the CGN device.

5. CGN device will look up the Couple table that formulated in the step c), use the source address of received packet (192.0.2.1:7000 in this example) to get the reflex IPv6 address of Peer B. It then will change the source address of the packet to the well-known public IPv6 relay address of CGN device, the destination address of this packet to the IPv6 reflex address of Peer B.

6. The translated IPv6 packet will be forwarded to the Peer B, processed by the Application Provider1's client located on it.

7. The return traffic will also be sent to the well-known IPv6 relay address/port of CGN device, converted by the CGN device, and sent back to the AP1 Client.

The AP1 client and Peer B will not notice the other end is located in different address families. The CGN device finishes all the network/ transport layer related translation work.

Such procedures are all same, regardless of the application using TCP or UDP transport, located in IPv4 environments or in IPv6 environments.

## 6. STUN Couple Command Definition

In order to let the CGN device to build one Couple item upon the request of Application Provider, it is needed to define one Couple message to transfer the related information.

### 6.1. Couple Opcode

The Couple request defines the relationship between two TCP or UDP half-connections, the translation rule that converts both the source address and destination address of pass through packet in both directions.

Couple Opcode: It defines how to bind two half-connections that ends at the CGN well-known transport address together. When CGN device receives the Couple request, it will create one map table item that includes the reflex IP address/port of both hosts that lies behind the NAT device and the protocol that the host will use to communicate.

When the CGN device receives the packet from one host, it will use the reflex source address/port to lookup the map table item; converts the source address/port of this packet to the well-known PCP server/ port and converts the destination address/port of this packet to the address/port that results from the map table lookup action.

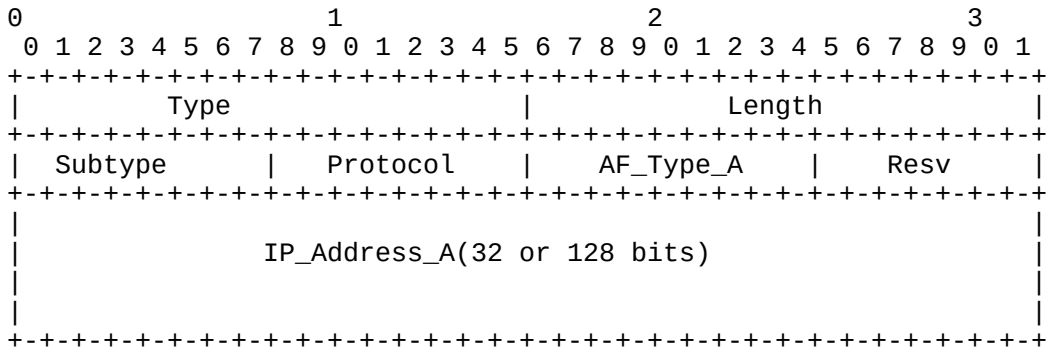
The converted packet will be routed to NAT side of the other host, NATed by the NAT device and then to the other host. The return packet will be delivered to the well-known IP address/port of CGN device and be converted in reverse accordingly.

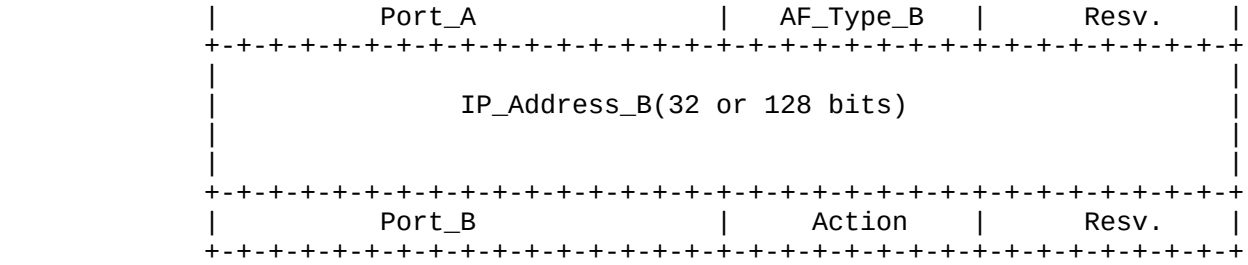
### 6.2. Couple Operation Packet Format

The Couple Opcode allows a TURNlite client to create a new explicit couple table on the CGN device (TURNLite Server), instructs the CGN device to accomplish the related translation work.

The following diagram shows the Opcode layout for the Couple Opcode request/response format.

Fig.7-1: Couple Opcode Request/Response Format






Type The value should be newly applied from IANA.

Length The total length of Couple packet.

Subtype This field will differentiate the Couple Request/Response packet. The value 0x01 indicates it's a Couple Request packet; The value 0x01 indicates it's a Couple Response packet.

Protocol Upper-layer protocol associated with this Opcode. Values are taken from the IANA protocol registry [proto_numbers]. For example, this field contains 6 (TCP) if the Opcode is intended to create a TCP mapping. This field contains 17(UDP) if the Opcode is intended to create a UDP mapping. The value 0 has a special meaning for 'all protocols'.

AF_Type_A The reflex address family of host A. It will be 4 when the host?s reflex address is IPv4 and will be 16 when the host?s reflex address is IPv6. It actually represents the length of following ?IP_Address_A? field.

IP_Address_A This field is the value of host A?s reflex address. Specially, in symmetric NAT environment, the reflex address is related to the well-known IP address/port of TURNLite server. For IPv6 host, the reflex address is often same as the local IPv6 address of host A.

AF_Type_B The reflex address family of host B. It will be 4 when the host?s reflex address is IPv4 and will be 16 when the host?s reflex address is IPv6. It actually represents the length of following ?IP_Address_B? field.

IP_Address_B This field is the value of host B?s reflex

address. Specially, in symmetric NAT environment, the reflex address is related to the well-known IP address/port of TURNLite server. For IPv6 host, the reflex address is often same as the local IPv6 address of host B.

Port_B This field is the value of host B?s reflex port. Specially, in symmetric NAT environment, the reflex address is related to the well-known IP address/port of TURNLite server.

Action It distinguishes the add/delete action that the TURNLite client wants to apply to the TURNLite server. The value 0x00 indicates deleting the related Couple item; 0x01 indicates adding the related Couple item.

Reserved Reserved byte, MUST be sent as 0 and MUST be ignored when received.

## 7. TURNLite Features

The TURNLite mainly aims to deal with the problems that are described in Section 3. TURNLite also supports some other good features as the following.

- Support of Symmetric NAT Traverse

  Due to every host communicate with the well-known relay transport address, there is no additional requirement for punching holes in the NAT devices, which is indispensable for the current TURN solution.

- Native Support of Host Mobility

  Since the host communicates with the well-known relay transport address, it will not sense the address change of the corresponding peer, then it support node mobility naturally. When the node move and application server detects it, the moving node only needs to resend the STUN binding command and report it to the application server. The application server just resends the Couple command to the TURNLite server; renew the Couple item within TURNLite server. There is no new command need to be defined.

- Less Relay Address Resource Consumption

It can conclude that the TURNLite solution simplifies the communication procedure apparently, and alleviate the burden of reserving and allocating large amount of relay address/port in TURN server.

o Simplified Procedures

Theoretically, TURNLite requires only two commands to accomplish the relay function, compared with over eight commands that required by TURN solution. Please see the figuire below for detail.

| | TURN Solution | TURNLite Solution |
|---|---|---|
| | 1. Binding | 1. Bindiing |
| Required | 2. Allocate | 2. Couple |
| Commands | 3. Send | |
| | 4. Data | |
| | 5. Channel Bind | |
| | 6. Connect | |
| | 7. ConnectionBind | |
| | 8. ConnectionAttempt | |

## 8. Deployment Consideration

The TURNLite Server can be deployed in distributed manner. The most appropriate devices for incorporating this function are the CGN devices that have been deployed distributed by the service provider. Each distributed TURNLite Server has one unique well-known IPv4/IPv6 transport address. These addresses can be obtained via the well-known FQDN of the TURNLite Server.

The application server can select the appropriate TURNLite Server based on the proximity of it with the communication hosts. The TURNLite Server selection policy can be controlled by the application server or service provider and is out of the scope of this document.

## 9. Security Considerations

The additional requirement of TURNLite is authenticating the couple operation from the TURNLite client to the TURNLite Server.

However, in theory the OAuth mechanism defined in current TURN solution could be reused in TURNLite. (Detailed considerations need further study.)

## 10. IANA Considerations

This draft requires IANA to allocate one new type value of STUN methods for the Couple command. (TBD)

## 11. Acknowledgements

This document was produced using the xml2rfc tool [RFC2629].

## 12. References

### 12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

### 12.2. Informative References

[I-D.ietf-tram-turnbis] Reddy, T., Johnston, A., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", draft-ietf-tram-turnbis-00 (work in progress), August 2014.

[I-D.wing-tram-turn-mobility] Wing, D., Patil, P., Reddy, T., and P. Martinsen, "Mobility with TURN", draft-wing-tram-turn-mobility-02 (work in progress), September 2014.

[RFC6062] Perreault, S. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, November 2010.

[RFC6156] Camarillo, G., Novo, O., and S. Perreault, "Traversal Using Relays around NAT (TURN) Extension for IPv6", RFC 6156, April 2011.

Authors' Addresses

Aijun Wang
China Telecom
China Telecom Coporation Limited Beijing Research Institute
No.118,Xizhimenneidajie,Xicheng District,Beijing, 100035
P.R. China

Email: wangaj@ctbri.com.cn

Bing Liu
Huawei Technologies
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: leo.liubing@huawei.com

TRAM
Internet-Draft
Intended status: Standards Track
Expires: March 20, 2015

D. Wing
P. Patil
T. Reddy
P. Martinsen
Cisco

September 16, 2014

# Mobility with TURN
# draft-wing-tram-turn-mobility-02

## Abstract

It is desirable to minimize traffic disruption caused by changing IP address during a mobility event. One mechanism to minimize disruption is to expose a shorter network path to the mobility event so only the local network elements are aware of the changed IP address but the remote peer is unaware of the changed IP address.

This draft provides such an IP address mobility solution using TURN. This is achieved by allowing a client to retain an allocation on the TURN server when the IP address of the client changes.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 20, 2015.

Table of Contents



## 1. Introduction

When moving between networks, the endpoint's IP address can change or (due to NAT) the endpoint's public IP address can change. Such a change of IP address breaks upper layer protocols such as TCP and RTP. Various techniques exist to prevent this breakage, all tied to making the endpoint's IP address static (e.g., Mobile IP, Proxy Mobile IP, LISP). Other techniques exist, which make the change in IP address agnostic to the upper layer protocol (e.g., SCTP). The mechanism described in this document are in that last category.

A TURN [RFC5766] server relays media packets and is used for a variety of purposes, including overcoming NAT and firewall traversal issues. The existing TURN specification does not permit a TURN

client to reuse an allocation across client IP address changes. Due to this, when the IP address of the client changes, the TURN client has to request for a new allocation, create permissions for the remote peer, create channels etc. In addition to notifying the remote peer of the address change, and punching new pinholes through any NAT/FW that might be on the path.

This specification describes a mechanism to seamlessly reuse allocations across client IP address changes without any of the hassles described above. A critical benefit of this technique is that the remote peer does not have to support mobility, or deal with any of the address changes. The client, that is subject to IP address changes, does all the work. The mobility technique works across and between network types (e.g., between 3G and wired Internet access), so long as the client can still access the TURN server. The technique should also work seamlessly when (D)TLS is used as a transport protocol for STUN. When there is a change in IP address, the client uses (D)TLS Session Resumption without Server-Side State as described in [RFC5077] to resume secure communication with the TURN server, using the changed client IP address.

## 2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

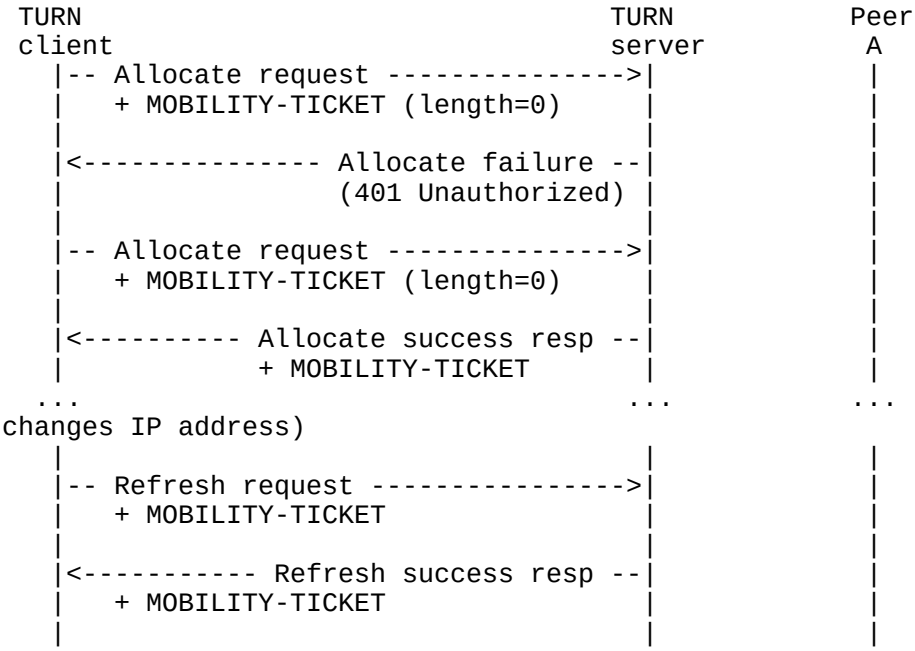
This note uses terminology defined in [RFC5245], and the following additional terminology:

## 3. Mobility using TURN

To achieve mobility, a TURN client should be able to retain an allocation on the TURN server across changes in the client IP address as a consequence of movement to other networks.

When the client sends the initial Allocate request to the TURN server, it will include a new STUN attribute MOBILITY-TICKET (with zero length value), which indicates that the client is capable of mobility and desires a ticket. The TURN server provisions a ticket that is sent inside the new STUN attribute MOBILITY-TICKET in the Allocate Success response to the client. The ticket will be used by the client when it wants to refresh the allocation but with a new client IP address and port. This ensures that an allocation can only be refreshed by the same client that allocated relayed transport address. When a client’s IP address changes due to mobility, it presents the previously obtained ticket in a Refresh Request to the TURN server. If the ticket is found to be valid, the TURN server

will retain the same relayed address/port for the new IP address/port allowing the client to continue using previous channel bindings -- thus, the TURN client does not need to obtain new channel bindings. Any data from external peer will be delivered by the TURN server to this new IP address/port of the client. The TURN client will continue to send application data to its peers using the previously allocated channelBind Requests.




## 3.1. Creating an Allocation

### 3.1.1. Sending an Allocate Request

In addition to the process described in Section 6.1 of [RFC5766], the client includes the MOBILITY-TICKET attribute with length 0. This indicates the client is a mobile node and wants a ticket.

### 3.1.2. Receiving an Allocate Request

In addition to the process described in Section 6.2 of [RFC5766], the server does the following:

If the MOBILITY-TICKET attribute is included, and has length zero, but TURN session mobility is forbidden by local policy, the server MUST reject the request with the new Mobility Forbidden error code. If the MOBILITY-TICKET attribute is included and has non-zero length

then the server MUST generate an error response with an error code of 400 (Bad Request). Following the rules specified in [RFC5389], if the server does not understand the MOBILITY-TICKET attribute, it ignores the attribute.

If the server can successfully process the request create an allocation, the server replies with a success response that includes a STUN MOBILITY-TICKET attribute. TURN server can store system internal data into the ticket that is encrypted by a key known only to the TURN server and sends the ticket in the STUN MOBILITY-TICKET attribute as part of Allocate success response. The ticket is opaque to the client, so the structure is not subject to interoperability concerns, and implementations may diverge from this format. An example for ticket construction is discussed in Appendix A. The client could be roaming across networks with different path MTU and from one address family to another (e.g. IPv6 to IPv4). The TURN server to support mobility must assume that the path MTU is unknown and the STUN message over IPv4 needs to be less than 548 bytes to avoid UDP fragmentation(Section 7.1 of [RFC5389]), it MUST ensure that the ticket length is restricted to fit within the 548 byte STUN message size. Clients MUST NOT examine the ticket under the assumption that it complies with this document.

### 3.1.3. Receiving an Allocate Success Response

In addition to the process described in Section 6.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the response. This attribute will be presented by the client to the server during a subsequent Refresh request to aid mobility.

### 3.1.4. Receiving an Allocate Error Response

If the client receives an Allocate error response with error code TBD (Mobility Forbidden), the error is processed as follows:

o TBD (Mobility Forbidden): The request is valid, but the server is refusing to perform it, likely due to administrative restrictions. The client considers the current transaction as having failed. The client MAY notify the user or operator and SHOULD NOT retry the same request with this server until it believes the problem has been fixed.

All other error responses must be handled as described in [RFC5766].

### 3.2. Refreshing an Allocation

#### 3.2.1. Sending a Refresh Request

If a client wants to refresh an existing allocation and update its time-to-expiry or delete an existing allocation, it will send a Refresh Request as described in Section 7.1 of [RFC5766]. If the client wants to retain the existing allocation in case of IP change, it will include the MOBILITY-TICKET attribute received in the Allocate Success response. If a Refresh transaction was previously made, the MOBILITY-TICKET attribute received in the Refresh Success response of the transaction must be used.

#### 3.2.2. Receiving a Refresh Request

In addition to the process described in Section 7.2 of [RFC5766], the server does the following:

If the STUN MOBILITY-TICKET attribute is included in the Refresh Request then the server will not retrieve the 5-tuple from the packet to identify an associated allocation. Instead the TURN server will decrypt the received ticket, verify the ticket’s validity and retrieve the 5-tuple allocation using the ticket. If this 5-tuple obtained does not identify an existing allocation then the server MUST reject the request with an error.

If the source IP address and port of the Refresh Request is different from the stored 5-tuple allocation, the TURN server proceeds with MESSAGE-INTEGRITY validation to identify the that it is the same user which had previously created the TURN allocation. If the above checks are not successful then server MUST reject the request with a 441 (Wrong Credentials) error.

If all of the above checks pass, the TURN server understands that the client has moved to a new network and acquired a new IP address. The source IP address of the request could either be the host transport address or server-reflexive transport address. The server then updates it’s 5-tuple with the new client IP address and port. TURN server calculates the ticket with the new 5-tuple and sends the new ticket in the STUN MOBILITY-TICKET attribute as part of Refresh Success response. The old ticket can only be used for the purposes of retransmission. If the client wants to refresh its allocation with a new server-reflexive transport address, it MUST use the new ticket. If the TURN server has not received a Refresh Request with STUN MOBILITY-TICKET attribute but receives Send indications or ChannelData messages from a client, the TURN server may discard or queue those Send indications or ChannelData messages (at its discretion). Thus, it is RECOMMENDED that the client avoid

transmitting a Send indication or ChannelData message until it has received an acknowledgement for the Refresh Request with STUN MOBILITY-TICKET attribute.

To accommodate for loss of Refresh responses, a server must retain the old STUN MOBILITY-TICKET attribute for a period of at least 30 seconds to be able recognize a retransmission of Refresh request with the old STUN MOBILITY-TICKET attribute from the client.

#### 3.2.3. Receiving a Refresh Response

In addition to the process described in Section 7.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the response. This attribute will be presented by the client to the server during a subsequent Refresh Request to aid mobility.

### 3.3. New STUN Attribute MOBILITY-TICKET

This attribute is used to retain an Allocation on the TURN server. It is exchanged between the client and server to aid mobility. The value of MOBILITY-TICKET is encrypted and is of variable-length.

### 3.4. New STUN Error Response Code

This document defines the following new error response code:

> Mobility Forbidden: Mobility request was valid but cannot be performed due to administrative or similar restrictions.

## 4. IANA Considerations

IANA is requested to add the following attributes to the STUN attribute registry [iana-stun],

- MOBILITY-TICKET (0x802E, in the comprehension-optional range)

and to add a new STUN error code "Mobility Forbidden" with the value 405 to the STUN Error Codes registry [iana-stun].

## 5. Implementation Status

[Note to RFC Editor: Please remove this section and reference to [RFC6982] prior to publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to

assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

### 5.1. open-sys

Organization: This is a public project, the full list of authors and contributors here: http://turnserver.open-sys.org/downloads/AUTHORS

Description: A mature open-source TURN server specs implementation (RFC 5766, RFC 6062, RFC 6156, etc) designed for high-performance applications, especially geared for WebRTC.

Implementation: http://code.google.com/p/rfc5766-turn-server/

Level of maturity: The Mobile ICE feature implementation can be qualified as "production" - it is well tested and fully implemented, but not widely used, yet..

Coverage: Fully implements MICE with TURN protocol.

Licensing: BSD: http://turnserver.open-sys.org/downloads/LICENSE

Implementation experience: MICE implementation is somewhat challenging for a multi-threaded performance-oriented application (because the mobile ticket information must be shared between the threads) but it is doable.

Contact: Oleg Moskalenko <mom040267@gmail.com>.

## 6. Security Considerations

TURN server MUST use strong encryption and integrity protection for the ticket to prevent an attacker from using a brute force mechanism to obtain the ticket’s contents or refreshing allocations. The

ticket MUST be constructed such that it has strong entropy to ensure nothing can be gleaned by looking at the ticket alone.

Security considerations described in [RFC5766] are also applicable to this mechanism.

## 7. Acknowledgements

Thanks to Alfred Heggestad, Lishitao, Sujing Zhou, Martin Thomson, Emil Ivov, Oleg Moskalenko and Brandon Williams for review and comments.

## 8. References

### 8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

### 8.2. Informative References

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[iana-stun] IANA, , "IANA: STUN Attributes", April 2011, <http://www.iana.org/assignments/stun-parameters/stun-parameters.xml>.

## Appendix A. Example ticket construction

The TURN server uses two different keys: one 128-bit key for Advance Encryption Standard (AES) in Cipher Block Chaining (CBC) mode (AES_128_CBC) and 256-bit key for HMAC-SHA-256-128 for integrity protection. The ticket can be structured as follows:

```
struct {
    opaque key_name[16];
    opaque iv[16];
    opaque state<0..2^16-1>;
    opaque mac[16];
} ticket;
```

Figure 1: Ticket Format

Here, key_name serves to identify a particular set of keys used to protect the ticket. It enables the TURN server to easily recognize tickets it has issued. The key_name should be randomly generated to avoid collisions between servers. One possibility is to generate new random keys and key_name every time the server is started.

The TURN state information (self-contained or handle) in encrypted_state is encrypted using 128-bit AES in CBC mode with the given IV. The MAC is calculated using HMAC-SHA-256-128 over key_name (16 octets)and IV (16 octets), followed by the length of the encrypted_state field (2 octets) and its contents (variable length).

## Authors' Addresses

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Prashanth Patil
Cisco Systems, Inc.
Bangalore
India

Email: praspati@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens vei 22
Lysaker, Akershus 1325
Norway

Email: palmarti@cisco.com