

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.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Notational Conventions	3
3. Mobility using TURN	3
3.1. Creating an Allocation	4
3.1.1. Sending an Allocate Request	4
3.1.2. Receiving an Allocate Request	4
3.1.3. Receiving an Allocate Success Response	5
3.1.4. Receiving an Allocate Error Response	5
3.2. Refreshing an Allocation	6
3.2.1. Sending a Refresh Request	6
3.2.2. Receiving a Refresh Request	6
3.2.3. Receiving a Refresh Response	7
3.3. New STUN Attribute MOBILITY-TICKET	7
3.4. New STUN Error Response Code	7
4. IANA Considerations	7
5. Implementation Status	7
5.1. open-sys	8
6. Security Considerations	8
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Appendix A. Example ticket construction	10
Authors' Addresses	10

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].

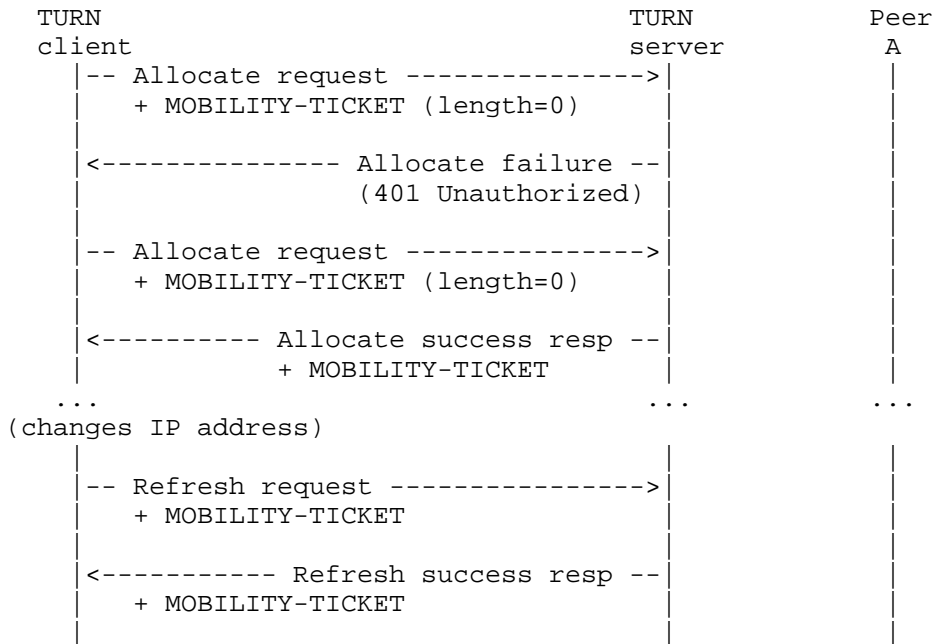
This note uses terminology defined in [RFC5245], and the following additional terminology:

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.

3.1.1.3. 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.

3.1.1.4. 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 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 its 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],

- o 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 Heggstad, Lishitao, Sujing Zhou, Martin Thomson, Emil Ivov, Oleg Moskalkenko 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