

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

P. Martinsen
Cisco
J. Uberti
Google
O. Moskalkenko
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

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. Creating an Allocation	3
2.1. Sending an Allocate Request	3
2.2. Receiving an Allocate Request	3
2.3. Receiving an Allocate Success Response	5
2.4. Receiving an Allocate Error Response	5
3. Refreshing an Allocation	5
3.1. Sending a Refresh Request	6
3.2. Receiving a Refresh Request	6
3.3. CreatePermission	6
3.3.1. Sending a CreatePermission Request	6
3.3.2. Receiving a CreatePermission Request	7
4. Channels	7
5. Implementation Status	7
5.1. open-sys	7
5.2. NATTools	8
6. Security Considerations	8
7. Acknowledgements	9
8. Normative References	9
Authors' Addresses	9

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 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/>