

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

B. Schwartz
J. Uberti
Google
March 5, 2015

TURN by name: an extension to TURN for contacting an endpoint by its DNS name.

[draft-schwartz-tram-turnbyname-00](#)

Abstract

When tunneling traffic through TURN, a client may sometimes desire to contact a remote endpoint that it knows by its DNS name, not its IP address. This document describes an extension to TURN that allows such a client to contact a named endpoint.

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 September 6, 2015.

Copyright Notice

Copyright (c) 2015 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.

Internet-Draft

TURN-BY-NAME

March 2015

Table of Contents

1.	Introduction	2
2.	New Address Family for DNS names	3
3.	Extension to the XOR-PEER-ADDRESS attribute format	3
4.	Changes to TURN server behavior	4
4.1.	Servers that do not support the extension	4
4.2.	Supported attributes	4
4.3.	Supported messages	4
4.4.	Name mapping storage	4
4.5.	Lookup behavior	5
4.6.	CreatePermission	6
4.6.1.	Implications of this permission model	6
4.7.	Send	7
4.8.	Channel Binding	7
4.8.1.	Implications of this channel binding model	8
4.9.	Receiving Data	9
4.9.1.	Implications of this data receipt model	9
5.	Changes to TURN client behavior	10
5.1.	When to use this extension	10
5.2.	Issuing Send, CreatePermission, and ChannelBind requests for DNS names	10
5.3.	Receiving Data indications	10
5.4.	Handling dynamic addressing	11
5.5.	Dual-stack behavior	11
6.	Examples	12
7.	Security Considerations	13
8.	IANA Considerations	14
9.	Acknowledgements	14
10.	References	14
10.1.	Normative References	14
10.2.	Informative References	14
	Authors' Addresses	15

[1.](#) Introduction

The TURN standard [[RFC5766](#)] extends STUN to allow proxying connections directly through the server. Clients send messages to the server in order to request the allocation of ports on the server, and identify the remote peers with whom they want to exchange packets. These remote peers are identified by an XOR-PEER-ADDRESS attribute, which includes the remote peer's IP address and port.

TURN is most commonly used as a component of an ICE [[RFC5245](#)] implementation, to allow communication between endpoints that each send their own transport address to the other in the form of an ICE candidate. These candidates are typically constructed using STUN, or

by direct interrogation of the network interfaces, so they normally contain IP addresses, although domain names are also allowed.

However, TURN is now attracting a wider range of use cases, especially on enterprise networks and in conjunction with WebRTC. Some use cases employ TURN as an "escape hatch" in an otherwise tightly restricted network, with the intention that users would tunnel much or all of their UDP traffic through the TURN server. On some restricted networks, DNS access is also restricted, which may prevent users from determining the IP address of a domain (e.g. an application-specified TURN server, for RETURN [[I-D.schwartz-rtcweb-return](#)], or an ICE candidate that contains a domain name) that they wish to contact through the escape-hatch TURN server.

Extending TURN to support named peers allows TURN to work for clients who are attempting to contact an endpoint by name, on networks where resolving those names is not otherwise possible.

[2.](#) New Address Family for DNS names

The Address Family 0x03 is defined to indicate that the specified address is a DNS name. This family is only permitted under certain circumstances, detailed below.

[3.](#) Extension to the XOR-PEER-ADDRESS attribute format

STUN/TURN attributes are Type-Length-Value encoded ([\[RFC5389\], Section 15](#)). For both of the existing Families, the attribute's encoded length is a known constant, because the length of the address is constant. For the newly defined Family 0x03, the length is variable, and the indicated length from the TLV encoding is necessary in order to parse the attribute.

The DNS name is transmitted in the X-Address field, and is encoded by the following procedure:

1. Define the "legacy transaction ID" as a 128-bit value consisting of the 32-bit magic cookie followed by the 96-bit transaction ID.
2. Define the DNS name as a standard dot-separated UTF-8 byte-string (not null-terminated).
3. Compute the encoded address (X-Address) by XOR'ing each byte of the DNS name with the corresponding byte of the legacy transaction ID.

- * If the DNS name is longer than 128 bits, the corresponding byte with which to XOR wraps around to the beginning of the legacy transaction ID.

This procedure is an extension of the encoding for families 0x01 and 0x02, so all three XOR-PEER-ADDRESS families can be encoded and parsed by a single procedure, without any special cases.

[4.](#) Changes to TURN server behavior

[4.1.](#) Servers that do not support the extension

Servers are NOT REQUIRED to support this extension. No change is required to servers that do not support the extension. Upon receiving a message containing an XOR-PEER-ADDRESS attribute with Family 0x03, existing compliant servers MUST reply with Error 440 (Address Family not Supported).

Servers that do support this extension MUST comply with the requirements that follow in this section.

[4.2.](#) Supported attributes

Address Family 0x03 is only permitted in the context of the XOR-PEER-ADDRESS attribute. All other attributes that use address families remain restricted to families 0x01 and 0x02. The server MUST respond with Error 440 (Address Family not Supported) when encountering this address family in an attribute where it is not supported.

[4.3.](#) Supported messages

The XOR-PEER-ADDRESS attribute may only have family 0x03 in the context of a CreatePermission, Send, Data, or ChannelBind message. If the server encounters this address family in the context of any other message type, it MUST respond with Error 440 (Address Family not Supported).

If a TURN server supports address family 0x03 in one of these messages, it MUST support it in all of these messages.

[4.4.](#) Name mapping storage

Baseline TURN servers must store two kinds of state for each Allocation: Permissions and Channel Bindings. This extension adds a third kind of state: Name Mappings. Each DNS Name Mapping consists of:

- o a DNS name

- o an IP address
- o a reference count, which is always either 1 or 2

Name Mappings do not have an expiration time, but the server MUST delete them if their reference count falls to zero. Like Permissions and Channel Bindings, Name Mappings are scoped to a single Allocation.

Each IP address appears in only one Name Mapping for an Allocation. The requirements for CreatePermission and ChannelBind are structured to maintain this invariant.

Server implementations SHOULD implement Name Mappings in a way that enables fast bidirectional lookup.

[4.5.](#) Lookup behavior

When a DNS name lookup is required, the server's behavior depends on the current Allocation. Each supported message is associated with an Allocation, whose address family is IPv4, IPv6 [[RFC6156](#)], or Both (via ADDITIONAL-ADDRESS-FAMILY [[I-D.ietf-tram-turnbis](#)]).

If the address family is IPv4, then the server MUST search for an A record for the name, and similarly if the address family is IPv6, the server MUST search for a AAAA record. The server MUST handle errors as follows:

- o If resolution fails due to a server error (e.g. DNS SERVFAIL), reply with error code 500 (Server Error).
- o If the resolution fails because there is no record of the required type (e.g. DNS NOERROR), respond with error code 443 (Peer Address Family Mismatch).
- o For all other DNS errors, return error code 447 (Connection timeout or failure).

The TURN server implementation MAY use a high-level DNS resolution API, such as `gethostbyname` or `getaddrinfo`, to perform the lookup.

If the Allocation has both address families, then it MUST look for an IPv6 address, and fall back to IPv4 only if a AAAA record is not found.

[4.6.](#) CreatePermission

In baseline TURN, each CreatePermission message creates or renews a Permission to send and receive messages to some specified IP address. With this extension, a Permission may indicate either an IP address or a DNS name. Both types of Permissions are subject to the same expiration policy.

At any given time, there is at most one Permission that specifies any IP address, or any DNS name, but there may be a Permission specifying a DNS name that resolves to an IP address that is specified in another Permission.

Upon receiving a CreatePermission message on an Allocation, the server MUST perform these steps:

1. If the CreatePermission message contains a peer address of family 0x01 or 0x02, create or update a Permission for the given address. (No change from baseline.)
2. If the CreatePermission message contains peer address of family 0x03:
 - A. Look for an existing Permission with the given DNS name. If one exists, refresh its expiration time and return success.
 - B. Otherwise, check if there is a Name Mapping for the DNS name.
 - i. If one exists, increment its reference count.
 - ii. Otherwise, perform a DNS lookup for the name. If it succeeds, add a DNS name mapping for the name and the resolved address, with reference count 1.
 - C. Install a new permission for the DNS name.

When a Permission containing a DNS name expires, the server MUST decrement the reference count on the Name Mapping for this DNS name, and delete the Name Mapping if its reference count falls to zero.

[4.6.1.](#) Implications of this permission model

As long as a permission is regularly refreshed with the same DNS name, the effective IP address will not change.

Permission refreshes for an IP address do not extend the lifetime of DNS resolutions to that address.

Permission requests for an IP address are not sufficient to allow Send requests to a DNS name that resolves to that IP address, and vice versa.

[4.7.](#) Send

Upon receiving a Send message on an Allocation, the server MUST perform these steps:

1. If the Send message contains a peer address of family 0x01 or 0x02, check for a Permission that indicates that IP address. (There will be at most one.) If a Permission matches, send the packet; otherwise silently drop it. (No change from baseline.)
2. If the Send message contains a peer address of family 0x03, check if there is a Permission for the given DNS name. (There will be at most one.) If one exists, send the packet to the IP address indicated for that DNS name in its Name Mapping; otherwise silently drop it.

[4.8.](#) Channel Binding

In baseline TURN, each ChannelBind message creates or renews a channel binding, which consists of a transport ID, a peer's IP address, and a port on that address. It also creates or renews a permission for the peer's IP address, exactly as if a CreatePermission message had been received for that IP address.

In this extension, each channel binding includes either an IP address or a DNS name.

Upon receiving a ChannelBind message on an Allocation, the server MUST perform these steps:

1. If the ChannelBind indicates a peer address of family 0x01 or 0x02
 - A. If a binding already exists with the specified transport ID, IP address, and port, refresh the binding.
 - B. If a binding already exists for the specified transport ID with a different or unspecified IP address or port, report Error 400 (Bad Request).
 - C. If a binding already exists with this port and this IP address, or a DNS name that maps to this IP address, report Error 400 (Bad Request) and include a CHANNEL-NUMBER

channel.

- D. Otherwise, create a binding.
 - E. Install or refresh a permission for the originally indicated peer IP address.
2. If the ChannelBind indicates a peer address of family 0x03
- A. If a binding already exists with the specified transport ID, DNS name, and port, refresh the binding, including the IP address.
 - B. Otherwise, resolve the DNS name to an IP address, using the name mapping table if it exists, and performing a DNS lookup only if no name mapping exists for this DNS name.
 - i. If a binding already exists for the specified transport ID with a different IP address or port, report Error 400 (Bad Request).
 - ii. If a binding already exists with a different transport ID, for this port, and this IP address or a DNS name that is mapped to this IP address, report Error 400 (Bad Request) and include a CHANNEL-NUMBER attribute that indicates the number of the conflicting channel.
 - C. Install a channel binding with the specified transport ID, DNS name, and port.
 - D. Increment the name mapping's reference count, or Install a new name mapping if one does not already exist for this DNS name.
 - E. Perform the steps required when receiving a CreatePermission message for this DNS name.

When a channel binding that indicates a DNS name expires, the server MUST decrement the reference count on the matching name mapping, and delete the mapping if the reference count falls to zero.

[4.8.1.](#) Implications of this channel binding model

As long as a channel is refreshed before it times out, it will continue to resolve to a constant address.

There can never be two channels bound to the same remote transport address. If that were possible, it would result in traffic amplification (sending each received packet to all matching channels) or other strange behaviors (e.g. selecting one arbitrary channel to receive the packet).

Each time a new channel is bound for a DNS name, it checks for a Name Mapping before doing any external resolution, so the resolved IP address is guaranteed to be consistent with the active Permission for this DNS name, if one exists. As a result, DNS resolution results can persist indefinitely within an Allocation, longer than the DNS TTL or any individual connection, if they are maintained by ChannelBind or CreatePermission calls to different ports on the same remote peer that overlap in time.

If two ChannelBind requests are received for the same port on two different DNS names that resolve to the same IP address, the second request will fail with a generic error code (400), but will also let the client know which existing channel to use instead. The same is true of collisions between IP and DNS channel binding requests.

Installing a channel binding to a DNS name also enables Send messages to the DNS name, but not to the resolved IP address.

[4.9.](#) Receiving Data

Upon receiving an incoming packet on an Allocation, the server **MUST** perform these steps:

1. Check if there is a channel binding to this source port and IP address, or a DNS name that is mapped to this IP address. (There will be at most one such channel.) If there is, let the channel handle the packet.
2. Otherwise, check if there is any DNS permission that is mapped to the source IP address. If there is, produce a Data message with that DNS name.
3. Otherwise, check if there is any IP permission that matches the source IP address. If there is, produce a Data message with the source IP address; otherwise discard the packet.

[4.9.1.](#) Implications of this data receipt model

If a name mapping exists for an IP address, all packets received from that address will be labeled with the DNS name, not the IP address.

Clients never learn the IP address for a DNS name unless they provoke a conflict, similar to the naming model used by SOCKS5 [[RFC1928](#)].

If a channel is bound for a port on a peer, all packets from that port will be routed to the channel exclusively.

[5.](#) Changes to TURN client behavior

Clients are NOT REQUIRED to support this extension. No change is required to existing clients. The requirements in this section only apply to clients that opt to support the extension.

[5.1.](#) When to use this extension

When the client receives a request to contact an endpoint that is identified by its DNS name, the client SHOULD attempt to use this extension to reach that endpoint, and SHOULD NOT attempt to perform a local DNS lookup for the name, so that connections may succeed even if the local DNS server fails to return a correct result.

If the TURN server responds with Error 440 (Address Family Not Supported), then the TURN client application SHOULD attempt to perform a local DNS lookup for the name, and retry the connection by IP address. (This functionality is logically separable from the TURN protocol itself, and might best be implemented by having a TURN client library that indicates the error, leaving the DNS lookup to be the responsibility of the application that uses the library.)

[5.2.](#) Issuing Send, CreatePermission, and ChannelBind requests for DNS names

When attempting to contact an endpoint by its DNS name, the client SHOULD transmit a CreatePermission or ChannelBind request whose XOR-PEER-ADDRESS attribute contains family 0x03, conveying the DNS name formatted as described above.

If the server responds with Error 440 (Address family not supported), then the client SHOULD abandon all requests using DNS, because the server does not support this extension.

If a ChannelBind request fails with Error 400, but includes a CHANNEL-NUMBER attribute, then that channel is already bound to the

remote transport address.

[5.3.](#) Receiving Data indications

Clients MAY send CreatePermission requests for both an IP address and a DNS name that maps to that IP address, and both requests will succeed. However, all Data messages from the remote peer will be marked as being received from the DNS name. Therefore, clients MUST

Schwartz & Uberti

Expires September 6, 2015

[Page 10]

Internet-Draft

TURN-BY-NAME

March 2015

NOT assume that replies from a Send to an IP address are labeled with that IP address.

[5.4.](#) Handling dynamic addressing

The IP address to which a DNS name resolves is not a constant. It may change occasionally due to address reassignment, or it may even change on every lookup, in the case of round-robin DNS.

The TURN server ensures that the IP address associated with a permission or channel binding does not change as long as the permission or binding is refreshed before it expires. Therefore, clients that need to send messages to a stable IP address MUST refresh their DNS name permissions and channel bindings even while they are not in use, to ensure that they do not expire and later resolve to a different IP address.

If the client has previously connected to a DNS name on an Allocation, and wishes to connect again to the same DNS name with an up-to-date IP address resolution, it SHOULD request a new Allocation, and connect to the DNS name on the new Allocation.

[5.5.](#) Dual-stack behavior

If a specific address family is not indicated for the remote endpoint, and the server does not support dual allocation (e.g. ADDITIONAL-ADDRESS-FAMILY [[I-D.ietf-tram-turnbis](#)]), then the client's behavior is implementation-defined. For example, when processing a request to send the first packet to a DNS name, the client MAY use an approach inspired by Happy Eyeballs [[RFC6555](#)]:

- o Create an Allocation for the system's preferred address family

(e.g. IPv6).

- o Attempt to connect to the DNS name on this Allocation using a ChannelBind message.
- * If the server replies with error code 443 (Peer Address Family Mismatch), immediately discard the Allocation and try again with an Allocation of the other family.
- * If a response message is received before some timeout (e.g. 300 ms), use this Allocation
- * If no response message is received before some timeout (e.g. 300 ms), attempt to connect using a new Allocation of the other address family, and use whichever Allocation receives a response first. Discard the other Allocation.

Schwartz & Uberti

Expires September 6, 2015

[Page 11]

Internet-Draft

TURN-BY-NAME

March 2015

6. Examples

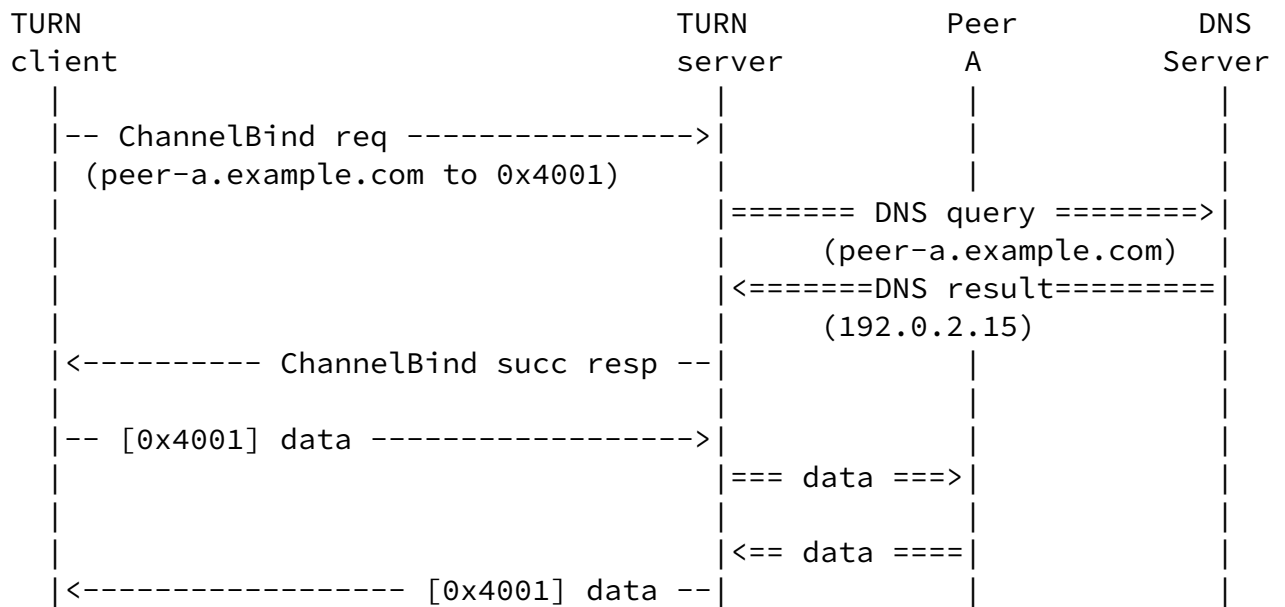
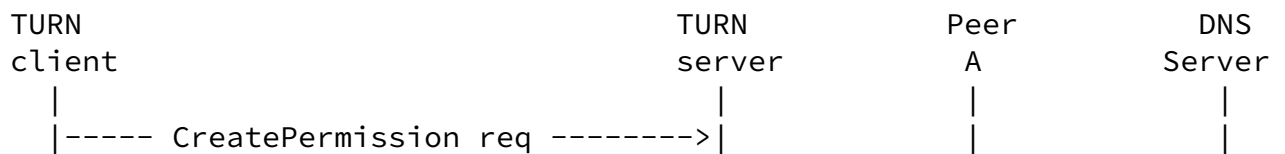


Figure 1: Using DNS names with ChannelBind



(peer-a.example.com)	
	===== DNS query =====> (peer-a.example.com)
	<===== DNS result =====
	(192.0.2.15)
<-- CreatePermission success resp --	
-- Send ind (peer-a.example.com) -->	
	=== data ===>
	<== data ==
<-- Data ind (peer-a.example.com) --	

Figure 2: Using DNS names with CreatePermission and Send

TURN client	TURN server	Peer A	DNS Server
----- CreatePermission req ----->			
(peer-a.example.com)			
	===== DNS query =====>		
	(peer-a.example.com)		
	<===== DNS result =====		
	(192.0.2.15)		
<-- CreatePermission success resp --			
----- ChannelBind req ----->			
(peer-a.example.com to 0x4001)			
<---- ChannelBind succ resp -----			
-- Send ind (peer-a.example.com) -->			
	=== data ===>		

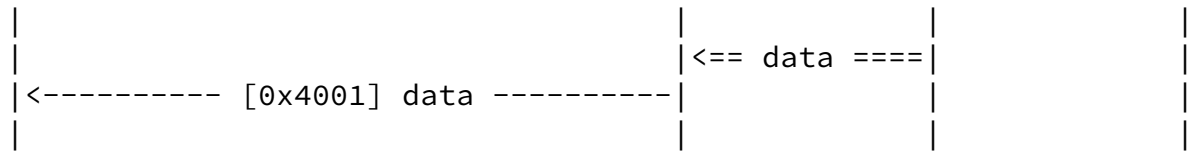


Figure 3: Sharing DNS names between CreatePermission and ChannelBind

7. Security Considerations

TURN servers that implement this specification can be made to parse arbitrary DNS records. They should make sure to use secure, well-tested DNS client implementations.

Clients can cause the TURN server to perform an arbitrary number of DNS lookups. Server implementations MAY limit the rate at which an individual client can trigger lookups, and return Error 508 (Insufficient Capacity) when a client exceeds the limit.

A malicious server could forward messages to the wrong IP address for a specified domain name, but this does not represent a change in security relative to the basic TURN standard.

To provide this functionality, the server is required to store a number of DNS Name Mappings that is at most the number of active permissions or channels. Implementers should take care to avoid resource leaks in the DNS mapping implementation, to maintain this bound.

8. IANA Considerations

This draft adds a new STUN address family, 0x03 (DNS name).

9. Acknowledgements

Thanks to Warren Kumari for his early review.

10. References

10.1. Normative 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-02](#) (work in progress), February
2015.
- [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.
- [RFC6156] Camarillo, G., Novo, O., and S. Perreault, "Traversal
Using Relays around NAT (TURN) Extension for IPv6", [RFC
6156](#), April 2011.

[10.2.](#) Informative References

- [I-D.schwartz-rtcweb-return]
Schwartz, B. and J. Uberti, "Recursively Encapsulated TURN
(RETURN) for Connectivity and Privacy in WebRTC", [draft-
schwartz-return-04](#) (work in progress), November 2014.
- [RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and
L. Jones, "SOCKS Protocol Version 5", [RFC 5766](#), March
1996.

Schwartz & Uberti Expires September 6, 2015 [Page 14]

Internet-Draft TURN-BY-NAME March 2015

- [RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with
Dual-Stack Hosts", [RFC 6555](#), April 2012.

Authors' Addresses

Benjamin M. Schwartz
Google, Inc.
111 8th Ave
New York, NY 10011
USA

Email: bemasc@webrtc.org

Justin Uberti
Google, Inc.
747 6th Street South
Kirkland, WA 98033
USA

Email: justin@uberti.name