Workgroup: Network Working Group
Internet-Draft:
draft-schinazi-masque-protocol-02
Published: September 10, 2020
Intended Status: Experimental
Expires: March 14, 2021
Authors: D. Schinazi
Google LLC

The MASQUE Protocol

Abstract

This document describes MASQUE (Multiplexed Application Substrate over QUIC Encryption). MASQUE is a framework that allows concurrently running multiple networking applications inside an HTTP/3 connection. For example, MASQUE can allow a QUIC client to negotiate proxying capability with an HTTP/3 server, and subsequently make use of this functionality while concurrently processing HTTP/3 requests and responses.

Discussion of this work is encouraged to happen on the MASQUE IETF mailing list masque@ietf.org or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/masque-drafts.

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 https://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 14, 2021.

Copyright Notice

Copyright (c) 2020 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 (https://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

This document describes MASQUE (Multiplexed Application Substrate over QUIC Encryption). MASQUE is a framework that allows concurrently running multiple networking applications inside an HTTP/3 connection (see [HTTP3]). For example, MASQUE can allow a QUIC client to negotiate proxying capability with an HTTP/3 server, and subsequently make use of this functionality while concurrently processing HTTP/3 requests and responses.

MASQUE Negotiation is performed using HTTP mechanisms, but MASQUE applications can subsequently leverage QUIC [QUIC] features without using HTTP.

Discussion of this work is encouraged to happen on the MASQUE IETF mailing list masque@ietf.org or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/masque-drafts.

1.1. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. MASQUE Negotiation

In order to negotiate the use of the MASQUE protocol, the client starts by sending a MASQUE request in the HTTP data of an HTTP POST request to "/.well-known/masque/initial". The client can use this to request specific MASQUE applications and advertise support for MASQUE extensions. The MASQUE server indicates support for MASQUE by sending an HTTP status code 200 response, and can use the data to inform the client of which MASQUE applications are now in use, and various configuration parameters.

Both the MASQUE negotiation initial request and its response carry a sequence of MASQUE applications, as shown in Figure 1:

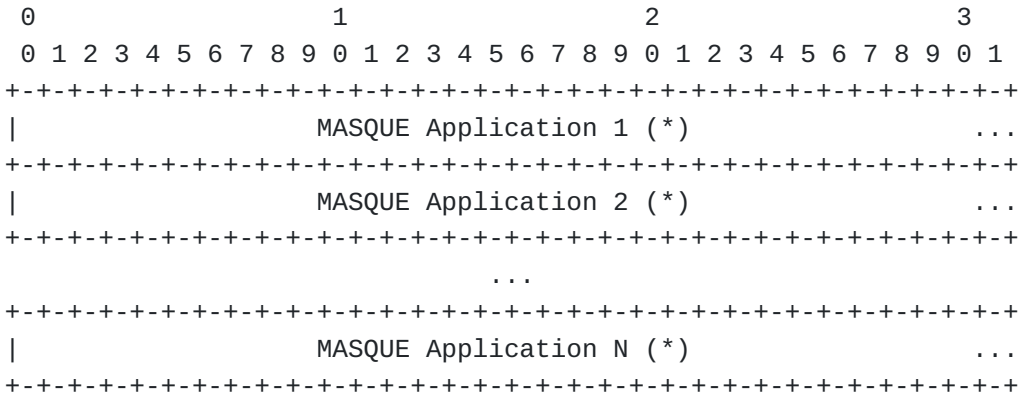

Figure 1: Sequence of MASQUE Applications

Each MASQUE Application is encoded as an (identifier, length, value) tuple, as shown in Figure 2:

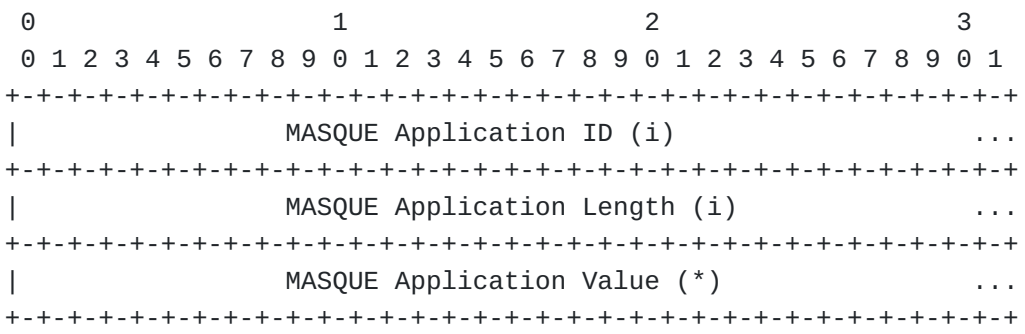

Figure 2: MASQUE Application Encoding

The MASQUE Application Length field contains the length of the MASQUE Application Value field. The contents of the MASQUE Application Value field are defined by its corresponding MASQUE application. When parsing, endpoints MUST ignore unknown MASQUE applications. A given MASQUE application ID MUST NOT appear twice in a given sequence of MASQUE applications.

3. MASQUE Applications

When the MASQUE server accepts the client's MASQUE initial request, it advertises support for MASQUE Applications, which will be multiplexed over this HTTP/3 connection.

3.1. HTTP Proxy

The client can make proxied HTTP requests through the server to other servers. In practice this will mean using the HTTP CONNECT method to establish a stream over which to run TLS to a different remote destination. The proxy applies back-pressure to streams in both directions.

3.1.1. HTTP Proxy Negotiation

Use of the HTTP Proxying MASQUE application is negotiated by sending the http_proxying (type 0x00) type-length-value during MASQUE negotiation. The length MUST be zero.

3.2. DNS over HTTPS

The client can send DNS queries using DNS over HTTPS [DOH] to the MASQUE server.

3.2.1. DNS over HTTPS Negotiation

Use of the DNS over HTTPS MASQUE application is negotiated by sending the dns_over_https (type 0x01) type-length-value during MASQUE negotiation. When sent by the client, the length MUST be

zero. When sent by the server, the value contains the DoH URI Template encoded as a non-null-terminated UTF-8 string.

3.3. QUIC Proxying

By leveraging QUIC client connection IDs, a MASQUE server can act as a QUIC proxy while only using one UDP port. To allow this, the MASQUE server informs the client of a required client connection ID length during negotiation. The client is then able to send proxied packets to the MASQUE server who will forward them on to the desired IP address and UDP port. Return packets are similarly forwarded in the opposite direction.

Compared to UDP proxying, this mode has the advantage of only requiring one UDP port to be open on the MASQUE server, and can lower the overhead on the link between client and MASQUE server by compressing connection IDs.

3.3.1. QUIC Proxy Compression

To reduce the overhead of proxying, QUIC Proxying leverages compression to elide the connection IDs on the link between the client and MASQUE server. This uses the concept of a compression context. Compression contexts are indexed using a datagram flow identifiers [H3DGRAM], and contain the tuple (client connection ID, server connection ID, server IP address, server port).

Any time and endpoint wants to send a proxied packet to its peer, it searches its list of compression contexts looking for one that matches the address, port and connection IDs from the proxied packet. If there was no match, the endpoint creates a new compression context and adds it to the list.

Compression contexts also carry a boolean value representing whether the context has been validated, which means that this endpoint is confident that its peer is aware if the given compression context. Compression contexts that were created by the peer start off validated, whereas locally-created ones are not validated until the endpoint receives a packet using that compression context, or an acknowledgement for a sent packet that uses the context.

The DATAGRAM frame [DGRAM] format below allows both endpoints to immediately start sending proxied QUIC packets using unvalidated compression contexts. Once contexts are vaidated, the server IP address, server port and the connection IDs can be ommited.

TODO: garbage collect obsolete compression contexts.

3.3.2. QUIC Proxy Negotiation

Use of the QUIC Proxying MASQUE application is negotiated by sending the quic_proxying (type 0x02) type-length-value during MASQUE negotiation.

When sent by the client, the value contains a single variable-length integer, called new_quic_proxy_compression_context_flow_id, that represents the DATAGRAM flow ID used to negotiate compression contexts.

When sent by the server, the value contains two variable-length integers, the DATAGRAM flow ID used to negotiate compression contexts (called new_quic_proxy_compression_context_flow_id), followed by the required connection ID length.

3.3.3. QUIC Proxy Encoding

Once negotiated, the QUIC Proxying MASQUE Application only uses DATAGRAM frames, whose content is shown in Figure 3 and Figure 4.

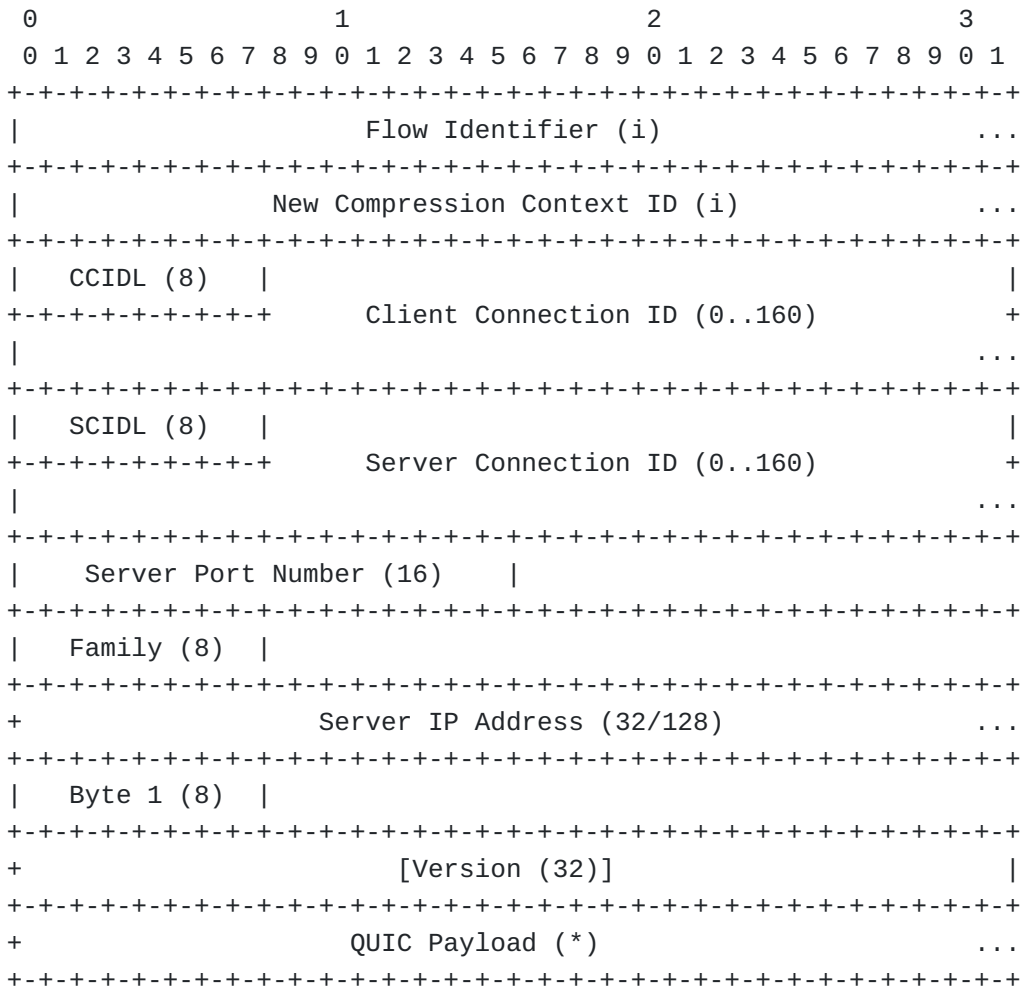

Figure 3: Unvalidated QUIC Proxy Datagram

Unvalidated QUIC Proxy DATAGRAM frames contain the following fields:

Flow Identifier: The flow identifier represents the compression context used by this frame. Unvalidated compression contexts use the new_quic_proxy_compression_context_flow_id value received during negotiation.

New Compression Context ID: The new Compression Context ID that this frame uses. The connection IDs and server IP address family, address, and port are associated with this context.

CCIDL: Length of the Client Connection ID field.

Client Connection ID: The client connection ID associated with this compression context.

SCIDL: Length of the Server Connection ID field.

Server Connection ID: The server connection ID associated with this compression context.

Server Port Number: The UDP port number used by the server.

Family: The IP address family of the Server IP Address field. Can be either 4 or 6.

Server IP Address: The IP address of the server. This field is 32 bits long if Family is 4 and 128 bits long if Family is 6.

Byte 1: The first byte of the QUIC packet being transferred.

Version: The QUIC version in the long header of the QUIC packet being transferred. This field is present if and only if the first bit of the Byte 1 field is set.

QUIC Payload: The contents of the QUIC packet being transmitted, starting after the last byte of last connection ID field.

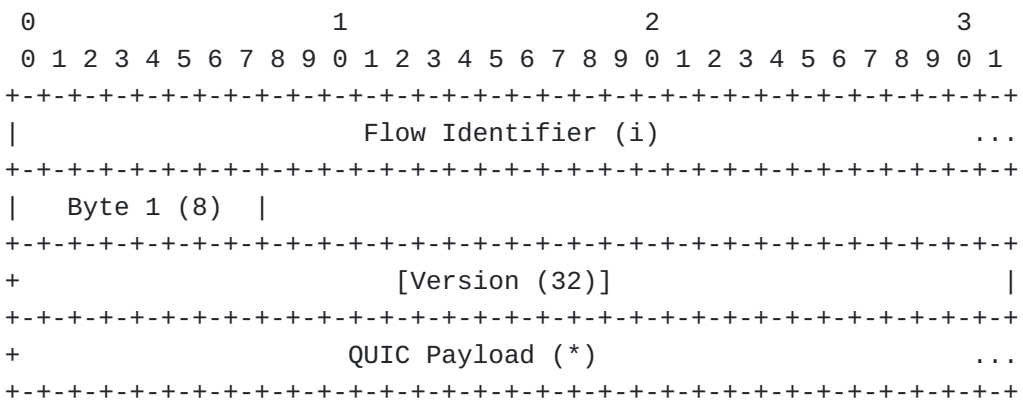

Figure 4: Validated QUIC Proxy Datagram

Validated QUIC Proxy DATAGRAM frames contain the following fields:

Flow Identifier: The flow identifier represents the compression context used by this frame. Calidated compression contexts MUST NOT use the new_quic_proxy_compression_context_flow_id value received during negotiation.

Byte 1: The first byte of the QUIC packet being transferred.

Version: The QUIC version in the long header of the QUIC packet being transferred. This field is present if and only if the first bit of the Byte 1 field is set.

QUIC Payload: The contents of the QUIC packet being transmitted, starting after the last byte of last connection ID field.

3.3.4. QUIC Proxy Routing

A MASQUE server keeps a mapping from client connection IDs to MASQUE clients, so that it can correctly route return packets. When a MASQUE server receives a QUIC Proxy DATAGRAM frame, it forwards it to the IP address and UDP port from the corresponding compression context. Additionally, the MASQUE server will ensure that the client connection ID from that compression context maps to that MASQUE client.

TODO: garbage collect this mapping from client connection ID to MASQUE client.

3.4. UDP Proxying

In order to support WebRTC to further servers, clients need a way to relay UDP onwards to a remote server. In practice for most widely deployed protocols other than DNS, this involves many datagrams over the same ports. Therefore this mechanism can compress the server's IP address and UDP port to reduce overhead.

3.4.1. UDP Proxy Compression

UDP Proxy leverages compression similarly to QUIC proxying, except that it only compresses the IP address and port, not QUIC connection IDs.

3.4.2. UDP Proxy Negotiation

Use of the UDP Proxying MASQUE application is negotiated by sending the udp_proxying (type 0x03) type-length-value during MASQUE negotiation.

The value contains a single variable-length integer, called new_udp_proxy_compression_context_flow_id, that represents the DATAGRAM flow ID used to negotiate compression contexts.

3.4.3. UDP Proxy Encoding

Once negotiated, the UDP Proxying MASQUE Application only uses DATAGRAM frames, whose content is shown in Figure 5 and Figure 6.

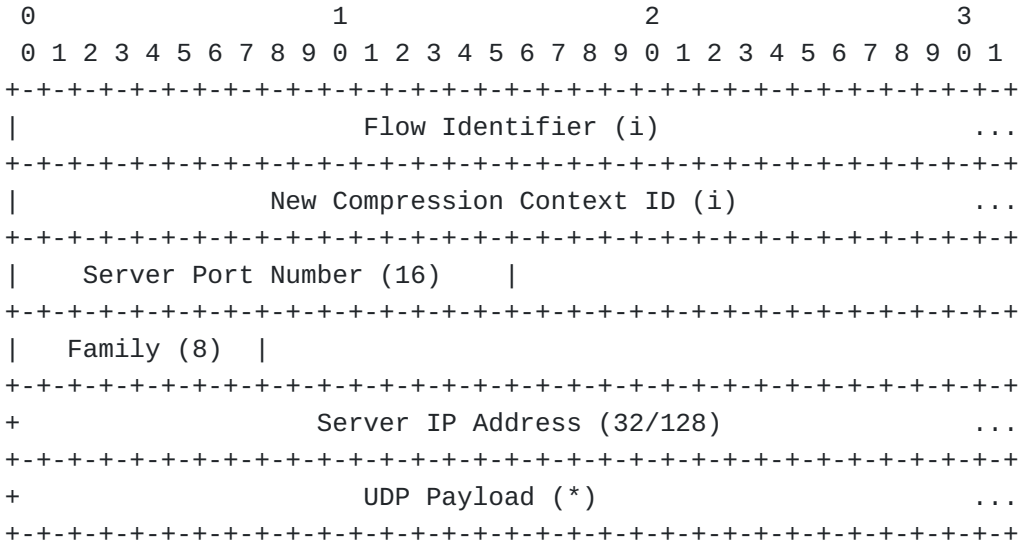

Figure 5: Unvalidated UDP Proxy Datagram

Unvalidated UDP Proxy DATAGRAM frames contain the following fields:

Flow Identifier: The flow identifier represents the compression context used by this frame. Unvalidated compression contexts use the new_udp_proxy_compression_context_flow_id value received during negotiation.

New Compression Context ID: The new Compression Context ID that this frame uses. The server IP address family, address, and port are associated with this context.

Server Port Number:
The UDP port number used by the server.

Family: The IP address family of the Server IP Address field. Can be either 4 or 6.

Server IP Address: The IP address of the server. This field is 32 bits long if Family is 4 and 128 bits long if Family is 6.

UDP Payload: The contents of the UDP packet being transmitted, starting after the last byte of the UDP checksum field.

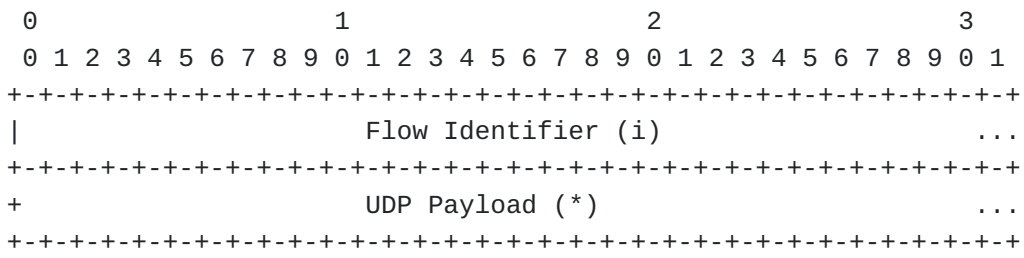

Figure 6: Validated UDP Proxy Datagram

Validated UDP Proxy DATAGRAM frames contain the following fields:

Flow Identifier: The flow identifier represents the compression context used by this frame. Calidated compression contexts MUST NOT use the new_udp_proxy_compression_context_flow_id value received during negotiation.

UDP Payload: The contents of the UDP packet being transmitted, starting after the last byte of the UDP checksum field.

3.5. IP Proxying

For the rare cases where the previous mechanisms are not sufficient, proxying can be performed at the IP layer. This would use a different DATAGRAM_ID and IP datagrams would be encoded inside it without framing.

3.5.1. IP Proxy Negotiation

Use of the IP Proxying MASQUE application is negotiated by sending the ip_proxying (type 0x04) type-length-value during MASQUE negotiation.

The value contains a single variable-length integer, called ip_proxy_flow_id, that represents the DATAGRAM flow ID used by IP Proxying DATAGRAM frames.

3.5.2. IP Proxy Encoding

Once negotiated, the IP Proxying MASQUE Application only uses DATAGRAM frames, whose content is shown in Figure 7.

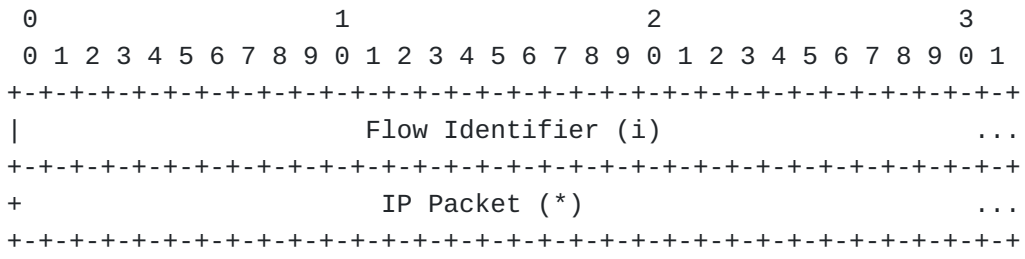

Figure 7: IP Proxy Datagram

IP Proxy DATAGRAM frames contain the following fields:

Flow Identifier: The flow identifier MUST be set to the ip_proxy_flow_id value received during negotiation.

IP Packet: The full IP packet, starting from the IP Version field.

3.6. Service Registration

MASQUE can be used to make a home server accessible on the wide area. The home server authenticates to the MASQUE server and registers a domain name it wishes to serve. The MASQUE server can then forward any traffic it receives for that domain name (by inspecting the TLS Server Name Indication (SNI) extension) to the home server. This received traffic is not authenticated and it allows non-modified clients to communicate with the home server without knowing it is not colocated with the MASQUE server.

To help obfuscate the home server, deployments can use Encrypted Server Name Indication [ESNI]. That will require the MASQUE server sending the cleartext SNI to the home server.

TODO: define the wire format for Service Registration.

4. Security Considerations

Here be dragons. TODO: slay the dragons.

5. IANA Considerations

5.1. MASQUE Well-Known URI

This document will request IANA to register the "/.well-known/masque/" URI (expert review) https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml.

5.2. MASQUE Applications Registry

This document will request IANA to create a new MASQUE Applications registry which governs a 62-bit space of MASQUE application types. This registry follows the same registration policy as the QUIC Transport Parameter Registry; see the IANA Considerations section of [QUIC].

The initial contents of this registry are shown in Table 1:

Value	Parameter Name	Specification
0x00	http_proxying	Section 3.1
0x01	dns_over_https	Section 3.2
0x02	quic_proxying	Section 3.3
0x03	udp_proxying	Section 3.4
0x04	ip_proxying	Section 3.5

Table 1: Initial MASQUE Applications Entries

6. References

6.1. Normative References

[DGRAM] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-datagram-01, August 24, 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-datagram-01.txt>.

[DOH] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[H3DGRAM] Schinazi, D., "Using QUIC Datagrams with HTTP/3", Work in Progress, Internet-Draft, draft-schinazi-quic-h3-datagram-04, April 16, 2020, <http://www.ietf.org/internet-drafts/draft-schinazi-quic-h3-datagram-04.txt>.

[HTTP3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-

http-29, June 9, 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-http-29.txt>.

[QUIC] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-29, June 9, 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-29.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

6.2. Informative References

[ESNI] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-07, June 1, 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-esni-07.txt>.

Acknowledgments

This proposal was inspired directly or indirectly by prior work from many people. The author would like to thank Nick Harper, Christian Huitema, Marcus Ihlar, Eric Kinnear, Mirja Kuehlewind, Brendan Moran, Lucas Pardue, Tommy Pauly, Zaheduzzaman Sarker, Ben Schwartz, and Christopher A. Wood for their input.

Author's Address

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043,
United States of America

Email: dschinazi.ietf@gmail.com