

Workgroup: MASQUE
Internet-Draft:
draft-ietf-masque-connect-udp-06
Published: 25 October 2021
Intended Status: Standards Track
Expires: 28 April 2022
Authors: D. Schinazi
Google LLC

UDP Proxying Support for HTTP

Abstract

This document describes how to proxy UDP over HTTP. Similar to how the CONNECT method allows proxying TCP over HTTP, this document defines a new mechanism to proxy UDP. It is built using HTTP Extended CONNECT.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the MASQUE WG mailing list (masque@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/masque/>.

Source for this draft and an issue tracker can be found at <https://github.com/ietf-wg-masque/draft-ietf-masque-connect-udp>.

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 28 April 2022.

Copyright Notice

Copyright (c) 2021 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](#)
 - [1.1. Conventions and Definitions](#)
- [2. Configuration of Clients](#)
- [3. HTTP Exchanges](#)
 - [3.1. Proxy Handling](#)
 - [3.2. HTTP Request over HTTP/1.1](#)
 - [3.3. HTTP Response over HTTP/1.1](#)
 - [3.4. HTTP Request over HTTP/2 and HTTP/3](#)
 - [3.5. HTTP Response over HTTP/2 and HTTP/3](#)
 - [3.6. Note About Draft Versions](#)
- [4. Encoding of Proxied UDP Packets](#)
- [5. Performance Considerations](#)
 - [5.1. MTU Considerations](#)
 - [5.2. Tunneling of ECN Marks](#)
- [6. Security Considerations](#)
- [7. IANA Considerations](#)
 - [7.1. HTTP Upgrade Token](#)
 - [7.2. Datagram Format Type](#)
- [8. References](#)
 - [8.1. Normative References](#)
 - [8.2. Informative References](#)
- [Acknowledgments](#)
- [Author's Address](#)

1. Introduction

This document describes how to proxy UDP over HTTP. Similar to how the CONNECT method (see [Section 9.3.6](#) of [\[SEMANTICS\]](#)) allows proxying TCP [\[TCP\]](#) over HTTP, this document defines a new mechanism to proxy UDP [\[UDP\]](#).

UDP Proxying supports all versions of HTTP and uses HTTP Datagrams [\[HTTP-DGRAM\]](#). When using HTTP/2 or HTTP/3, UDP proxying uses HTTP Extended CONNECT as described in [\[EXT-CONNECT2\]](#) and [\[EXT-CONNECT3\]](#). When using HTTP/1.x, UDP proxying uses HTTP Upgrade as defined in [Section 7.8](#) of [\[SEMANTICS\]](#).

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.

In this document, we use the term "proxy" to refer to the HTTP server that opens the UDP socket and responds to the UDP proxying request. If there are HTTP intermediaries (as defined in [Section 3.7](#) of [[SEMANTICS](#)]) between the client and the proxy, those are referred to as "intermediaries" in this document.

Note that, when the HTTP version in use does not support multiplexing streams (such as HTTP/1.1), any reference to "stream" in this document represents the entire connection.

2. Configuration of Clients

Clients are configured to use UDP Proxying over HTTP via an URI Template [[TEMPLATE](#)]. The URI template MUST contain exactly two variables: "target_host" and "target_port". Examples are shown below:

```
https://masque.example.org/{target_host}/{target_port}/  
https://proxy.example.org:4443/masque?h={target_host}&p={target_port}  
https://proxy.example.org:4443/masque{?target_host,target_port}
```

Figure 1: URI Template Examples

Since the original HTTP CONNECT method allowed conveying the target host and port but not the scheme, proxy authority, path, nor query, there exist proxy configuration interfaces that only allow the user to configure the proxy host and the proxy port. Client implementations of this specification that are constrained by such limitations MUST use the default template which is defined as: "https://\$PROXY_HOST:\$PROXY_PORT/{target_host}/{target_port}/" where \$PROXY_HOST and \$PROXY_PORT are the configured host and port of the proxy respectively. Proxy deployments SHOULD use the default template to facilitate interoperability with such clients.

3. HTTP Exchanges

This document defines the "connect-udp" HTTP Upgrade Token. "connect-udp" uses the Capsule Protocol as defined in [[HTTP-DGRAM](#)].

A "connect-udp" request requests that the recipient establish a tunnel over a single HTTP stream to the destination target server

identified by the "target_host" and "target_port" variables of the URI template (see [Section 2](#)). If the request is successful, the proxy commits to converting received HTTP Datagrams into UDP packets and vice versa until the tunnel is closed. Tunnels are commonly used to create an end-to-end virtual connection, which can then be secured using QUIC [[QUIC](#)] or another protocol running over UDP.

When sending its UDP proxying request, the client SHALL perform URI template expansion to determine the path and query of its request. target_host supports using DNS names, IPv6 literals and IPv4 literals. Note that this URI template expansion requires using pct-encoding, so for example if the target_host is "2001:db8::42", it will be encoded in the URI as "2001%3Adb8%3A%3A42".

A payload within a UDP proxying request message has no defined semantics; a UDP proxying request with a non-empty payload is malformed.

Responses to UDP proxying requests are not cacheable.

3.1. Proxy Handling

Upon receiving a UDP proxying request, the recipient proxy extracts the "target_host" and "target_port" variables from the URI it has reconstructed from the request headers, and establishes a tunnel by directly opening a UDP socket to the requested target.

Unlike TCP, UDP is connection-less. The proxy that opens the UDP socket has no way of knowing whether the destination is reachable. Therefore it needs to respond to the request without waiting for a packet from the target. However, if the target_host is a DNS name, the proxy MUST perform DNS resolution before replying to the HTTP request. If DNS resolution fails, the proxy MUST fail the request and SHOULD send details using the Proxy-Status header [[PROXY-STATUS](#)].

Proxies can use connected UDP sockets if their operating system supports them, as that allows the proxy to rely on the kernel to only send it UDP packets that match the correct 5-tuple. If the proxy uses a non-connected socket, it MUST validate the IP source address and UDP source port on received packets to ensure they match the client's request. Packets that do not match MUST be discarded by the proxy.

The lifetime of the socket is tied to the request stream. The proxy MUST keep the socket open while the request stream is open. If a proxy is notified by its operating system that its socket is no longer usable, it MUST close the request stream. Proxies MAY choose to close sockets due to a period of inactivity, but they MUST close the request stream before closing the socket. Proxies that close

sockets after a period of inactivity SHOULD NOT use a period lower than two minutes, see [Section 4.3](#) of [\[BEHAVE\]](#).

A successful response (as defined in [Section 3.3](#) and [Section 3.5](#)) indicates that the proxy has opened a socket to the requested target and is willing to proxy UDP payloads. Any response other than a successful response indicates that the request has failed, and the client MUST therefore abort the request.

3.2. HTTP Request over HTTP/1.1

When using HTTP/1.1, a UDP proxying request will meet the following requirements:

- *the method SHALL be "CONNECT".
- *the request-target SHALL use absolute-form (see [Section 3.2.2](#) of [\[MESSAGING\]](#)).
- *the request SHALL include a single Host header containing the origin of the proxy.
- *the request SHALL include a single "Connection" header with value "Upgrade".
- *the request SHALL include a single "Upgrade" header with value "connect-udp".

For example, if the client is configured with URI template "https://proxy.example.org/{target_host}/{target_port}/" and wishes to open a UDP proxying tunnel to target 192.0.2.42:443, it could send the following request:

```
CONNECT https://proxy.example.org/192.0.2.42/443/ HTTP/1.1
Host: proxy.example.org
Connection: upgrade
Upgrade: connect-udp
```

Figure 2: Example HTTP Request over HTTP/1.1

3.3. HTTP Response over HTTP/1.1

The proxy SHALL indicate a successful response by replying with the following requirements:

- *the HTTP status code on the response SHALL be 101 (Switching Protocols).

*the response SHALL include a single "Connection" header with value "Upgrade".

*the response SHALL include a single "Upgrade" header with value "connect-udp".

*the response SHALL NOT include any Transfer-Encoding or Content-Length header fields.

If any of these requirements are not met, the client MUST treat this proxying attempt as failed and abort the connection.

For example, the proxy could respond with:

```
HTTP/1.1 101 Switching Protocols
Connection: upgrade
Upgrade: connect-udp
```

Figure 3: Example HTTP Response over HTTP/1.1

3.4. HTTP Request over HTTP/2 and HTTP/3

When using HTTP/2 [H2] or HTTP/3 [H3], UDP proxying requests use HTTP pseudo-headers with the following requirements:

*The ":method" pseudo-header field SHALL be "CONNECT".

*The ":protocol" pseudo-header field SHALL be "connect-udp".

*The ":authority" pseudo-header field SHALL contain the authority of the proxy.

*The ":path" and ":scheme" pseudo-header fields SHALL NOT be empty. Their values SHALL contain the scheme and path from the URI template after the URI template expansion process has been completed.

A UDP proxying request that does not conform to these restrictions is malformed (see [Section 8.1.2.6](#) of [H2]).

For example, if the client is configured with URI template "https://proxy.example.org/{target_host}/{target_port}/" and wishes to open a UDP proxying tunnel to target 192.0.2.42:443, it could send the following request:

```
HEADERS
:method = CONNECT
:protocol = connect-udp
:scheme = https
:path = /192.0.2.42/443/
:authority = proxy.example.org
```

Figure 4: Example HTTP Request over HTTP/2

3.5. HTTP Response over HTTP/2 and HTTP/3

The proxy SHALL indicate a successful response by replying with any 2xx (Successful) HTTP status code, without any Transfer-Encoding or Content-Length header fields.

If any of these requirements are not met, the client MUST treat this proxying attempt as failed and abort the request.

For example, the proxy could respond with:

```
HEADERS
:status = 200
```

Figure 5: Example HTTP Response over HTTP/2

3.6. Note About Draft Versions

[[RFC editor: please remove this section before publication.]]

In order to allow implementations to support multiple draft versions of this specification during its development, we introduce the "connect-udp-version" header. When sent by the client, it contains a list of draft numbers supported by the client (e.g., "connect-udp-version: 0, 2"). When sent by the proxy, it contains a single draft number selected by the proxy from the list provided by the client (e.g., "connect-udp-version: 2"). Sending this header is RECOMMENDED but not required.

4. Encoding of Proxied UDP Packets

UDP packets are encoded using HTTP Datagrams [[HTTP-DGRAM](#)] with the UDP_PAYLOAD HTTP Datagram Format Type (see value in [Section 7.2](#)). When using the UDP_PAYLOAD HTTP Datagram Format Type, the payload of a UDP packet (referred to as "data octets" in [[UDP](#)]) is sent unmodified in the "HTTP Datagram Payload" field of an HTTP Datagram.

In order to use HTTP Datagrams, the client will first decide whether or not it will attempt to use HTTP Datagram Contexts and then

register its context ID (or lack thereof) using the corresponding registration capsule, see [[HTTP-DGRAM](#)].

When sending a registration capsule using the "Datagram Format Type" set to UDP_PAYLOAD, the "Datagram Format Additional Data" field SHALL be empty. Servers MUST NOT register contexts using the UDP_PAYLOAD HTTP Datagram Format Type. Clients MUST NOT register more than one context using the UDP_PAYLOAD HTTP Datagram Format Type. Endpoints MUST NOT close contexts using the UDP_PAYLOAD HTTP Datagram Format Type. If an endpoint detects a violation of any of these requirements, it MUST abort the stream.

Clients MAY optimistically start sending proxied UDP packets before receiving the response to its UDP proxying request, noting however that those may not be processed by the proxy if it responds to the request with a failure, or if the datagrams are received by the proxy before the request.

Extensions to this mechanism MAY define new HTTP Datagram Format Types in order to use different semantics or encodings for UDP payloads.

5. Performance Considerations

Proxies SHOULD strive to avoid increasing burstiness of UDP traffic: they SHOULD NOT queue packets in order to increase batching.

When the protocol running over UDP that is being proxied uses congestion control (e.g., [[QUIC](#)]), the proxied traffic will incur at least two nested congestion controllers. This can reduce performance but the underlying HTTP connection MUST NOT disable congestion control unless it has an out-of-band way of knowing with absolute certainty that the inner traffic is congestion-controlled.

If a client or proxy with a connection containing a UDP proxying request stream disables congestion control, it MUST NOT signal ECN support on that connection. That is, it MUST mark all IP headers with the Not-ECT codepoint. It MAY continue to report ECN feedback via ACK_ECN frames, as the peer may not have disabled congestion control.

When the protocol running over UDP that is being proxied uses loss recovery (e.g., [[QUIC](#)]), and the underlying HTTP connection runs over TCP, the proxied traffic will incur at least two nested loss recovery mechanisms. This can reduce performance as both can sometimes independently retransmit the same data. To avoid this, HTTP/3 datagrams SHOULD be used.

5.1. MTU Considerations

When using HTTP/3 with the QUIC Datagram extension [[DGRAM](#)], UDP payloads are transmitted in QUIC DATAGRAM frames. Since those cannot be fragmented, they can only carry payloads up to a given length determined by the QUIC connection configuration and the path MTU. If a proxy is using QUIC DATAGRAM frames and it receives a UDP payload from the target that will not fit inside a QUIC DATAGRAM frame, the proxy SHOULD NOT send the UDP payload in a DATAGRAM capsule, as that defeats the end-to-end unreliability characteristic that methods such as Datagram Packetization Layer Path MTU Discovery (DPLPMTUD) depend on [[RFC8899](#)]. In this scenario, the proxy SHOULD drop the UDP payload and send an ICMP "Packet Too Big" message to the target [[RFC4443](#)].

5.2. Tunneling of ECN Marks

UDP proxying does not create an IP-in-IP tunnel, so the guidance in [[RFC6040](#)] about transferring ECN marks between inner and outer IP headers does not apply. There is no inner IP header in UDP proxying tunnels.

Note that UDP proxying clients do not have the ability in this specification to control the ECN codepoints on UDP packets the proxy sends to the server, nor can proxies communicate the markings of each UDP packet from server to proxy.

A UDP proxy MUST ignore ECN bits in the IP header of UDP packets received from the server, and MUST set the ECN bits to Not-ECT on UDP packets it sends to the server. These do not relate to the ECN markings of packets sent between client and proxy in any way.

6. Security Considerations

There are significant risks in allowing arbitrary clients to establish a tunnel to arbitrary servers, as that could allow bad actors to send traffic and have it attributed to the proxy. Proxies that support UDP proxying SHOULD restrict its use to authenticated users.

Because the CONNECT method creates a TCP connection to the target, the target has to indicate its willingness to accept TCP connections by responding with a TCP SYN-ACK before the proxy can send it application data. UDP doesn't have this property, so a UDP proxy could send more data to an unwilling target than a CONNECT proxy. However, in practice denial of service attacks target open TCP ports so the TCP SYN-ACK does not offer much protection in real scenarios. Proxies MUST NOT introspect the contents of UDP payloads as that would lead to ossification of UDP-based protocols by proxies.

7. IANA Considerations

7.1. HTTP Upgrade Token

This document will request IANA to register "connect-udp" in the HTTP Upgrade Token Registry maintained at <<https://www.iana.org/assignments/http-upgrade-tokens>>.

Value: connect-udp

Description: Proxying of UDP Payloads.

Expected Version Tokens: None.

Reference: This document.

7.2. Datagram Format Type

This document will request IANA to register UDP_PAYLOAD in the "HTTP Datagram Format Types" registry established by [[HTTP-DGRAM](#)].

Type	Value	Specification
UDP_PAYLOAD	0xff6f00	This Document

Table 1: Registered Datagram Format Type

8. References

8.1. 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-06, 5 October 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-quic-datagram-06>>.
- [EXT-CONNECT2] McManus, P., "Bootstrapping WebSockets with HTTP/2", RFC 8441, DOI 10.17487/RFC8441, September 2018, <<https://www.rfc-editor.org/rfc/rfc8441>>.
- [EXT-CONNECT3] Hamilton, R., "Bootstrapping WebSockets with HTTP/3", Work in Progress, Internet-Draft, draft-ietf-httpbis-h3-websockets-00, 9 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-h3-websockets-00>>.
- [H2] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI

10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/rfc/rfc7540>>.

[H3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34>>.

[HTTP-DGRAM] Schinazi, D. and L. Pardue, "Using Datagrams with HTTP", Work in Progress, Internet-Draft, draft-ietf-masque-h3-datagram-05, 25 October 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-masque-h3-datagram-05>>.

[MESSAGING] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP/1.1", Work in Progress, Internet-Draft, draft-ietf-httpbis-messaging-19, 12 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-messaging-19>>.

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/rfc/rfc9000>>.

[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/rfc/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/rfc/rfc8174>>.

[SEMANTICS] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-19, 12 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-19>>.

[TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/rfc/rfc793>>.

[TEMPLATE] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/

RFC6570, March 2012, <<https://www.rfc-editor.org/rfc/rfc6570>>.

[UDP] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/rfc/rfc768>>.

8.2. Informative References

[BEHAVE] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <<https://www.rfc-editor.org/rfc/rfc4787>>.

[PROXY-STATUS] Nottingham, M. and P. Sikora, "The Proxy-Status HTTP Response Header Field", Work in Progress, Internet-Draft, draft-ietf-httpbis-proxy-status-08, 13 October 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-proxy-status-08>>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/rfc/rfc4443>>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/rfc/rfc6040>>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <<https://www.rfc-editor.org/rfc/rfc8899>>.

Acknowledgments

This document is a product of the MASQUE Working Group, and the author thanks all MASQUE enthusiasts for their contributions. This proposal was inspired directly or indirectly by prior work from many people. In particular, the author would like to thank Eric Rescorla for suggesting to use an HTTP method to proxy UDP. Thanks to Lucas Pardue for their inputs on this document.

Author's Address

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043,

United States of America

Email: dschinazi.ietf@gmail.com