Workgroup: MASQUE
Internet-Draft:
draft-ietf-masque-connect-udp-04
Published: 12 July 2021
Intended Status: Standards Track
Expires: 13 January 2022
Authors: D. Schinazi
Google LLC

The CONNECT-UDP HTTP Method

Abstract

This document describes the CONNECT-UDP HTTP method. CONNECT-UDP is similar to the HTTP CONNECT method, but it uses UDP instead of TCP.

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 13 January 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

This document describes the CONNECT-UDP HTTP method. CONNECT-UDP is similar to the HTTP CONNECT method (see section 4.3.6 of [RFC7231]), but it uses UDP [UDP] instead of TCP [TCP].

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 CONNECT-UDP request. If there are HTTP intermediaries (as defined in Section 2.3 of [RFC7230]) between the client and the proxy, those are referred to as "intermediaries" in this document.

2. Supported HTTP Versions

The CONNECT-UDP method is defined for all versions of HTTP. UDP payloads are sent using HTTP Datagrams [HTTP-DGRAM]. Note that, when the HTTP version in use does not support multiplexing streams (such as HTTP/1.1), then any reference to "stream" in this document is meant to represent the entire connection.

3. The CONNECT-UDP Method

The CONNECT-UDP method requests that the recipient establish a tunnel over a single HTTP stream to the destination origin server identified by the request-target and, if successful, thereafter restrict its behavior to blind forwarding of packets, in both directions, 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.

The request-target of a CONNECT-UDP request is a URI [RFC3986] which uses the "masque" scheme and an immutable path of "/". For example:

```
CONNECT-UDP masque://target.example.com:443/ HTTP/1.1
Host: target.example.com:443
```

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

* The ":method" pseudo-header field is set to "CONNECT-UDP".

* The ":scheme" pseudo-header field is set to "masque".

* The ":path" pseudo-header field is set to "/".

* The ":authority" pseudo-header field contains the host and port to connect to (similar to the authority-form of the request-target of CONNECT requests; see [RFC7230], Section 5.3).

A CONNECT-UDP request that does not conform to these restrictions is malformed (see [H2], Section 8.1.2.6).

The recipient proxy establishes a tunnel by directly opening a UDP socket to the request-target. Any 2xx (Successful) response indicates that the proxy has opened a socket to the request-target and is willing to proxy UDP payloads. Any response other than a successful response indicates that the tunnel has not yet been formed.

A proxy MUST NOT send any Transfer-Encoding or Content-Length header fields in a 2xx (Successful) response to CONNECT-UDP. A client MUST

treat a response to CONNECT-UDP containing any Content-Length or Transfer-Encoding header fields as malformed.

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

Responses to the CONNECT-UDP method are not cacheable.

4. Encoding of Proxied UDP Packets

UDP packets are encoded using HTTP Datagrams [HTTP-DGRAM]. 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 CONNECT-UDP client will first decide whether or not to use HTTP Datagram Contexts and then register its context ID (or lack thereof) using the corresponding registration capsule, see [HTTP-DGRAM].

Since HTTP Datagrams require prior negotiation (for example, in HTTP/3 it is necessary to both send and receive the H3_DATAGRAM SETTINGS Parameter), clients MUST NOT send any HTTP Datagrams until they have established support on a given connection. If negotiation of HTTP Datagrams fails (for example if an HTTP/3 SETTINGS frame was received without the H3_DATAGRAM SETTINGS Parameter), the client MUST consider its CONNECT-UDP request as failed.

The proxy that is creating the UDP socket to the destination responds to the CONNECT-UDP request with a 2xx (Successful) response, and indicates it supports HTTP Datagrams by sending the corresponding registration capsule.

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

Extensions to CONNECT-UDP MAY leverage the "Context Extensions" field of registration capsules in order to negotiate different semantics or encoding for UDP payloads.

5. Proxy Handling

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 CONNECT-UDP request without waiting for a TCP SYN-ACK.

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 CONNECT-UDP request. Packets that do not match MUST be discarded by the proxy.

The lifetime of the socket is tied to the CONNECT-UDP stream. The proxy MUST keep the socket open while the CONNECT-UDP stream is open. Proxies MAY choose to close sockets due to a period of inactivity, but they MUST close the CONNECT-UDP stream before closing the socket.

6. 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 CONNECT-UDP 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.

6.1. Tunneling of ECN Marks

CONNECT-UDP 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 CONNECT-UDP tunnels.

Note that CONNECT-UDP 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 CONNECT-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.

7. 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 CONNECT-UDP 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 CONNECT-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.

8. IANA Considerations

8.1. HTTP Method

This document will request IANA to register "CONNECT-UDP" in the HTTP Method Registry (IETF review) maintained at <https://www.iana.org/assignments/http-methods>.

Method Name	Safe	Idempotent	Reference
CONNECT-UDP	no	no	This document

8.2. URI Scheme Registration

This document will request IANA to register the URI scheme "masque".

The syntax definition below uses Augmented Backus-Naur Form (ABNF) [RFC5234]. The definitions of "host" and "port" are adopted from [RFC3986]. The syntax of a MASQUE URI is:

```
masque-URI = "masque:" "//" host ":" port "/"
```

The "host" and "port" component MUST NOT be empty, and the "port" component MUST NOT be 0.

9. References

9.1. Normative References

[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>.

[HTTP-DGRAM] Schinazi, D. and L. Pardue, "Using Datagrams with HTTP", Work in Progress, Internet-Draft, draft-ietf-masque-h3-datagram-03, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-h3-datagram-03>.

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/rfc/rfc7230>.

[RFC7231] "*** BROKEN REFERENCE ***".

[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>.

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

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

9.2. Informative References

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

Acknowledgments

This document is a product of the MASQUE Working Group, and the author thanks all MASQUE enthusiasts for their contibutions. 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