Workgroup: Network Working Group
Internet-Draft:
draft-ietf-masque-connect-udp-01
Published: 12 December 2020
Intended Status: Standards Track
Expires: 15 June 2021
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 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/ietf-wg-masque/draft-ietf-masque-connect-udp.

Discussion Venues

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

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 15 June 2021.

Copyright Notice

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].

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/ietf-wg-masque/draft-ietf-masque-connect-udp.

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. When the HTTP version used runs over QUIC [QUIC], UDP payloads can be sent over QUIC DATAGRAM frames [DGRAM]. Otherwise they are sent on the stream where the CONNECT-UDP request was made. 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 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. Note that the CONNECT-UDP stream is used to convey UDP packets, but they are not semantically part of the request or response themselves.

Responses to the CONNECT-UDP method are not cacheable.

4. Datagram Encoding of Proxied UDP Packets

When the HTTP connection supports HTTP/3 datagrams [H3DGRAM], UDP packets can be encoded using QUIC DATAGRAM frames. This support is ascertained by checking the received value of the H3_DATAGRAM SETTINGS Parameter.

If the client has both sent and received the H3_DATAGRAM SETTINGS Parameter with value 1 on this connection, it SHOULD attempt to use HTTP/3 datagrams. This is accomplished by requesting a datagram flow identifier from the flow identifier allocation service [H3DGRAM]. That service generates an even flow identifier, and the client sends it to the proxy by using the "Datagram-Flow-Id" header; see [H3DGRAM]. A CONNECT-UDP request with an odd flow identifier is malformed.

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 datagram encoding by echoing the "Datagram-Flow-Id" header. Once the client has received the "Datagram-Flow-Id" header on the successful response, it knows that it can use the HTTP/3 datagram encoding to send proxied UDP packets for this particular request. It then encodes the payload of UDP datagrams into the payload of HTTP/3 datagrams. Is the CONNECT-UDP response does not carry the "Datagram-Flow-Id" header, then the datagram encoding is not available for this request. A CONNECT-UDP response that carries the "Datagram-Flow-Id" header but with a different flow identifier than the one sent on the request is malformed.

When the proxy processes a new CONNECT-UDP request, it MUST ensure that the datagram flow identifier is not equal to flow identifiers

from other requests: if it is, the proxy MUST reject the request with a 4xx (Client Error) status code. Extensions MAY weaken or remove this requirement.

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 without echoing the "Datagram-Flow-Id" header, or if the datagrams arrive before the CONNECT-UDP request.

Note that a proxy can send the H3_DATAGRAM SETTINGS Parameter with a value of 1 while disabling datagrams on a particular request by not echoing the "Datagram-Flow-Id" header. If the proxy does this, it MUST NOT treat receipt of datagrams as an error, because the client could have sent them optimistically before receiving the response. In this scenario, the proxy MUST discard those datagrams.

Extensions to CONNECT-UDP MAY leverage parameters on the "Datagram-Flow-Id" header (parameters are defined in Section 3.1.2 of [STRUCT-HDR]). Proxies MUST NOT echo parameters on the "Datagram-Flow-Id" header if it does not understand their semantics.

5. Stream Encoding of Proxied UDP Packets

If HTTP/3 datagrams are not supported, the stream is used to convey UDP payloads, by using the following format (using the notation from the "Notational Conventions" section of [QUIC]):

```
CONNECT-UDP Stream Chunk {
  CONNECT-UDP Stream Chunk Type (i) = 0x00,
  UDP Payload Length (i),
  UDP Payload (..),
}
```

Figure 1: CONNECT-UDP Stream Chunk Format

CONNECT-UDP Stream Chunk Type: A variable-length integer indicating the Type of the CONNECT-UDP Stream Chunk, set to 0x00 to indicate a UDP Payload.

UDP Payload Length: The length of the UDP Payload field following this field.

UDP Payload: The payload of the UDP datagram.

The bidirectional stream that the CONNECT-UDP request was sent on is a sequence of CONNECT-UDP Stream Chunks. The CONNECT-UDP Stream Chunk Type is designed to allow future extensibility. Endpoints that

receive a chunk with an unknown CONNECT-UDP Stream Chunk Type MUST silently skip over that chunk.

6. 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.

7. HTTP Intermediaries

HTTP/3 DATAGRAM flow identifiers are specific to a given HTTP/3 connection. However, in some cases, an HTTP request may travel across multiple HTTP connections if there are HTTP intermediaries involved; see Section 2.3 of [RFC7230].

Intermediaries that support both CONNECT-UDP and HTTP/3 datagrams MUST negotiate flow identifiers separately on the client-facing and server-facing connections. This is accomplished by having the intermediary parse the "Datagram-Flow-Id" header on all CONNECT-UDP requests it receives, and sending the same value in the "Datagram-Flow-Id" header on the response. The intermediary then ascertains whether it can use datagrams on the server-facing connection. If they are supported (as indicated by the H3_DATAGRAM SETTINGS parameter), the intermediary uses its own flow identifier allocation service to allocate a flow identifier for the server-facing connection, and waits for the server's reply to see if the server sent the "Datagram-Flow-Id" header on the response. The intermediary then translates datagrams between the two connections by using the flow identifier specific to that connection. An intermediary MAY also choose to use datagrams on only one of the two connections, and translate between datagrams and streams.

8. 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.

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.

9. 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.

10. IANA Considerations

10.1. 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

10.2. 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.

10.3. Stream Chunk Type Registration

This document will request IANA to create a "CONNECT-UDP Stream Chunk Type" registry. This registry governs a 62-bit space, and follows the registration policy for QUIC registries as defined in [QUIC]. In addition to the fields required by the QUIC policy, registrations in this registry MUST include the following fields:

Type: A short mnemonic for the type.

Description: A brief description of the type semantics, which MAY be a summary if a specification reference is provided.

The initial contents of this registry are:

Value	Type	Description	Reference
0x00	UDP_PACKET	Payload of UDP packet	This document

Each value of the format 37 * N + 23 for integer values of N (that is, 23, 60, 97, ...) are reserved; these values MUST NOT be assigned by IANA and MUST NOT appear in the listing of assigned values.

11. 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, 24 August 2020,

<http://www.ietf.org/internet-drafts/draft-ietf-quic-datagram-01.txt>.

[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/info/rfc7540>.

[H3DGRAM] Schinazi, D., "Using QUIC Datagrams with HTTP/3", Work in Progress, Internet-Draft, draft-schinazi-masque-h3-datagram-01, 12 December 2020, <http://www.ietf.org/internet-drafts/draft-schinazi-masque-h3-datagram-01.txt>.

[QUIC] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-32, 20 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-32.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>.

[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/info/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/info/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/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC

7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[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>.

[STRUCT-HDR] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-header-structure-19, 3 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-header-structure-19.txt>.

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

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

Acknowledgments

This proposal was inspired directly or indirectly by prior work from many people. 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