Workgroup: MASQUE
Internet-Draft:
draft-pauly-masque-quic-proxy-04
Published: 2 September 2022
Intended Status: Experimental
Expires: 6 March 2023
Authors: T. Pauly
Apple Inc.
D. Schinazi
Google LLC

QUIC-Aware Proxying Using HTTP

Abstract

This document defines an extension to UDP Proxying over HTTP that adds specific optimizations for proxied QUIC connections. This extension allows a proxy to reuse UDP 4-tuples for multiple connections. It also defines a mode of proxying in which QUIC short header packets can be forwarded using an HTTP/3 proxy rather than being re-encapsulated and re-encrypted.

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/tfpauly/quic-proxy.

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 6 March 2023.

Copyright Notice

Table of Contents

1. Introduction

UDP Proxying over HTTP [CONNECT-UDP] defines a way to send datagrams through an HTTP proxy, where UDP is used to communicate between the proxy and a target server. This can be used to proxy QUIC connections [QUIC], since QUIC runs over UDP datagrams.

This document uses the term "target" to refer to the server that a client is accessing via a proxy. This target may be an origin hosting content, or another proxy.

This document extends the UDP proxying protocol to add signalling about QUIC Connection IDs. QUIC Connection IDs are used to identify QUIC connections in scenarios where there is not a strict bidirectional mapping between one QUIC connection and one UDP 4-tuple (pairs of IP addresses and ports). A proxy that is aware of Connection IDs can reuse UDP 4-tuples between itself and a target for multiple proxied QUIC connections.

Awareness of Connection IDs also allows a proxy to avoid re-encapsulation and re-encryption of proxied QUIC packets once a connection has been established. When this functionality is present, the proxy can support two modes for handling QUIC packets:

1. Tunnelled, in which client <-> target QUIC packets are encapsulated inside client <-> proxy QUIC packets. These packets use multiple layers of encryption and congestion control. QUIC long header packets MUST use this mode. QUIC short header packets MAY use this mode. This is the default mode for UDP proxying.

2. Forwarded, in which client <-> target QUIC packets are sent directly over the client <-> proxy UDP socket. These packets are only encrypted using the client-target keys, and use the client-target congestion control. This mode MUST only be used for QUIC short header packets.

Forwarding is defined as an optimization to reduce CPU processing on clients and proxies, as well as avoiding MTU overhead for packets on the wire. This makes it suitable for deployment situations that otherwise relied on cleartext TCP proxies, which cannot support QUIC and have inferior security and privacy properties.

The properties provided by the forwarding mode are as follows:

*All packets sent between the client and the target traverse through the proxy device.

*The target server cannot know the IP address of the client solely based on the proxied packets the target receives.

*Observers of either or both of the client <-> proxy link and the proxy <-> target are not able to learn more about the client <-> target communication than if no proxy was used.

It is not a goal of forwarding mode to prevent correlation between client <-> proxy and the proxy <-> target packets from an entity that can observe both links. See Section 8 for further discussion.

Both clients and proxies can unilaterally choose to disable forwarded mode for any client <-> target connection.

The forwarding mode of this extension is only defined for HTTP/3 [HTTP3] and not any earlier versions of HTTP.

QUIC proxies only need to understand the Header Form bit, and the connection ID fields from packets in client <-> target QUIC connections. Since these fields are all in the QUIC invariants header [INVARIANTS], QUIC proxies can proxy all versions of QUIC.

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.

1.2. Terminology

This document uses the following terms:

- Client: the client of all QUIC connections discussed in this document.

- Proxy: the endpoint that responds to the UDP proxying request.

- Target: the server that a client is accessing via a proxy.

- Client <-> Proxy HTTP stream: a single HTTP stream established from the client to the proxy.

- Socket: a UDP 4-tuple (local IP address, local UDP port, remote IP address, remote UDP port). In some implementations, this is referred to as a "connected" socket.

- Client-facing socket: the socket used to communicate between the client and the proxy.

- Target-facing socket: the socket used to communicate between the proxy and the target.

- Client Connection ID: a QUIC Connection ID that is chosen by the client, and is used in the Destination Connection ID field of packets from the target to the client.

*Target Connection ID: a QUIC Connection ID that is chosen by the target, and is used in the Destination Connection ID field of packets from the client to the target.

*Virtual Target Connection ID: a fake QUIC Connection ID that is chosen by the proxy that the client MUST use when sending QUIC packets in forwarding mode.

1.3. Virtual Target Connection ID

QUIC allows each endpoint of a connection to choose the connection IDs they receive with. Load balancing strategies such as those described in [QUIC-LB] may choose to take advantage of this by encoding routing information in the connection ID. When operating in forwarding mode, clients send QUIC packets destined for the Target directly to the Proxy. Since these packets are generated using the Target Connection ID, load balancers may not be able to route packets to the correct Proxy.

The Virtual Target Connection ID is a connection ID chosen by the Proxy that the Client uses when sending forwarded mode packets. The Proxy replaces the Virtual Target Connection ID with the Target Connection ID prior to forwarding the packet to the Target. This is only necessary in the Client->Target direction because the Proxy is otherwise the only receiver of QUIC packets with connection IDs it did not generate.

Clients and Proxies not implementing forwarding mode do not need to consider the Virtual Target Connection ID since all Client->Target datagrams will be encapsulated within the Client<->Proxy connection.

2. Required Proxy State

In the methods defined in this document, the proxy is aware of the QUIC Connection IDs being used by proxied connections, along with the sockets used to communicate with the client and the target. Tracking Connection IDs in this way allows the proxy to reuse target-facing sockets for multiple connections and support the forwarding mode of proxying.

QUIC packets can be either tunnelled within an HTTP proxy connection using HTTP Datagram frames [HTTP-DGRAM], or be forwarded directly alongside an HTTP/3 proxy connection on the same set of IP addresses and UDP ports. The use of forwarded mode requires the consent of both the client and the proxy.

In order to correctly route QUIC packets in both tunnelled and forwarded modes, the proxy needs to maintain mappings between several items. There are three required unidirectional mappings, described below.

2.1. Stream Mapping

Each client <-> proxy HTTP stream MUST be mapped to a single target-facing socket.

(Client <-> Proxy HTTP Stream) => Target-facing socket

Multiple streams can map to the same target-facing socket, but a single stream cannot be mapped to multiple target-facing sockets.

This mapping guarantees that any HTTP Datagram using a stream sent from the client to the proxy in tunnelled mode can be sent to the correct target.

2.2. Virtual Target Connection ID Mapping

Each pair of Virtual Target Connection ID and client-facing socket MUST map to a single target-facing socket and Target Connection ID.

(Client-facing socket + Virtual Target Connection ID)
 => (Target-facing socket + Target Connection ID)

Multiple pairs of Connection IDs and client-facing sockets can map to the same target-facing socket.

This mapping guarantees that any QUIC packet containing the Virtual Target Connection ID sent from the client to the proxy in forwarded mode can be sent to the correct target with the correct Target Connection ID. Thus, a proxy that does not allow forwarded mode does not need to maintain this mapping.

2.3. Client Connection ID Mappings

Each pair of Client Connection ID and target-facing socket MUST map to a single stream on a single client <-> proxy HTTP stream. Additionally, when supporting forwarding mode, the pair of Client Connection ID and target-facing socket MUST map to a single client-facing socket.

(Target-facing socket + Client Connection ID) => (Client <-> Proxy HTTP
(Target-facing socket + Client Connection ID) => Client-facing socket

Multiple pairs of Connection IDs and target-facing sockets can map to the same HTTP stream or client-facing socket.

These mappings guarantee that any QUIC packet sent from a target to the proxy can be sent to the correct client, in either tunnelled or forwarded mode. Note that this mapping becomes trivial if the proxy always opens a new target-facing socket for every client request

with a unique stream. The mapping is critical for any case where target-facing sockets are shared or reused.

2.4. Detecting Connection ID Conflicts

In order to be able to route packets correctly in both tunnelled and forwarded mode, proxies check for conflicts before creating a new mapping. If a conflict is detected, the proxy will reject the client's request, as described in Section 5.

Two sockets conflict if and only if all members of the 4-tuple (local IP address, local UDP port, remote IP address, and remote UDP port) are identical.

Two Connection IDs conflict if and only if one Connection ID is equal to or a prefix of another. For example, a zero-length Connection ID conflicts with all connection IDs. This definition of a conflict originates from the fact that QUIC short headers do not carry the length of the Destination Connection ID field, and therefore if two short headers with different Destination Connection IDs are received on a shared socket, one being a prefix of the other prevents the receiver from identifying which mapping this corresponds to.

The proxy treats two mappings as being in conflict when a conflict is detected for all elements on the left side of the mapping diagrams above.

Since very short Connection IDs are more likely to lead to conflicts, particularly zero-length Connection IDs, a proxy MAY choose to reject all requests for very short Connection IDs as conflicts, in anticipation of future conflicts.

3. Connection ID Capsule Types

Proxy awareness of QUIC Connection IDs relies on using capsules ([HTTP-DGRAM]) to signal the addition and removal of Client and Target Connection IDs.

Note that these capsules do not register contexts. QUIC packets are encoded using HTTP Datagrams with the context ID set to zero as defined in [CONNECT-UDP].

The capsules used for QUIC-aware proxying allow a client to register connection IDs with the proxy, and for the proxy to acknowledge or reject the connection ID mappings.

The REGISTER_CLIENT_CID and REGISTER_TARGET_CID capsule types (see Section 9.2 for the capsule type values) allow a client to inform the proxy about a new Client Connection ID or a new Target

Connection ID, respectively. These capsule types MUST only be sent by a client.

The ACK_CLIENT_CID and ACK_TARGET_CID capsule types (see Section 9.2 for the capsule type values) are sent by the proxy to the client to indicate that a mapping was successfully created for a registered connection ID as well as provide the Virtual Target Connection ID that may be used in forwarding mode. These capsule types MUST only be sent by a proxy.

The CLOSE_CLIENT_CID and CLOSE_TARGET_CID capsule types (see Section 9.2 for the capsule type values) allow either a client or a proxy to remove a mapping for a connection ID. These capsule types MAY be sent by either a client or the proxy. If a proxy sends a CLOSE_CLIENT_CID without having sent an ACK_CLIENT_CID, or if a proxy sends a CLOSE_TARGET_CID without having sent an ACK_TARGET_CID, it is rejecting a Connection ID registration.

All capsule types except for ACK_TARGET_CID are formatted as follows:

```
Connection ID Capsule {
  Type (i) = 0xffe200..0xffe202, 0xffe204..0xffe205
  Length (i),
  Connection ID (0..2040),
}
```

Figure 1: Connection ID Capsule Format

Connection ID: A connection ID being registered or acknowledged, which is between 0 and 255 bytes in length. The length of the connection ID is implied by the length of the capsule. Note that in QUICv1, the length of the Connection ID is limited to 20 bytes, but QUIC invariants allow up to 255 bytes.

The ACK_TARGET_CID capsule type includes the Virtual Target Connection ID and a Stateless Reset Token.

```
Virtual Target Connection ID Capsule {
  Type (i) = 0xffe203,
  Length (i)
  Connection ID Length (i)
  Connection ID (0..2040),
  Virtual Target Connection ID Length (i)
  Virtual Target Connection ID (0..2040),
  Stateless Reset Token Length (i),
  Stateless Reset Token (..),
}
```

Figure 2: Virtual Target Connection ID Capsule Format

Connection ID Length The length of the connection ID being acknowledged, which is between 0 and 255. Note that in QUICv1, the length of the Connection ID is limited to 20 bytes, but QUIC invariants allow up to 255 bytes.

Connection ID A connection ID being acknowledged whose length is equal to Connection ID Length. This is the real Target Connection ID.

Virtual Target Connection ID Length The length of the connection ID being provided to the client. This must be a valid connection ID length for the QUIC version used in the client<->proxy QUIC connection. When forwarding mode is not negotiated, the length MUST be zero.

Virtual Target Connection ID The Proxy-chosen connection ID that the client MUST use when sending packets in forwarding mode. The proxy rewrites forwarding mode packets to contain the correct Target Connection ID prior to forwarding them on to the Target.

Stateless Reset Token Length The length of the stateless reset token that may be sent by the proxy in response to forwarded mode packets in order to reset the Client<->Target QUIC connection. When forwarding mode is not negotiated, the length MUST be zero. Proxies choosing not to support stateless resets MAY set the length to zero. Clients receiving a zero-length stateless reset token MUST ignore it.

Stateless Reset Token A Stateless Reset Token provided by the Proxy to the Client allowing the Proxy to reset the Client<->Target connection in response to Client->Target forwarding mode packets.

4. Client Request Behavior

A client initiates UDP proxying via a CONNECT request as defined in [CONNECT-UDP]. Within its request, it includes the "Proxy-QUIC-Forwarding" header to indicate whether or not the request should support forwarding. If this header is not included, the client MUST NOT send any connection ID capsules.

The "Proxy-QUIC-Forwarding" is an Item Structured Header [RFC8941]. Its value MUST be a Boolean. Its ABNF is:

```
Proxy-QUIC-Forwarding = sf-boolean
```

If the client wants to enable QUIC packet forwarding for this request, it sets the value to "?1". If it doesn't want to enable forwarding, but instead only provide information about QUIC

Connection IDs for the purpose of allowing the proxy to share a target-facing socket, it sets the value to "?0".

If the proxy supports QUIC-aware proxying, it will include the "Proxy-QUIC-Forwarding" header in successful HTTP responses. The value indicates whether or not the proxy supports forwarding. If the client does not receive this header in responses, the client SHALL assume that the proxy does not understand how to parse Connection ID capsules, and MUST NOT send any Connection ID capsules.

The client sends a REGISTER_CLIENT_CID capsule whenever it advertises a new Client Connection ID to the target, and a REGISTER_TARGET_CID capsule when it has received a new Target Connection ID for the target. Note that the initial REGISTER_CLIENT_CID capsule MAY be sent prior to receiving an HTTP response from the proxy.

4.1. New Proxied Connection Setup

To initiate QUIC-aware proxying, the client sends a REGISTER_CLIENT_CID capsule containing the initial Client Connection ID that the client has advertised to the target.

If the mapping is created successfully, the client will receive a ACK_CLIENT_CID capsule that contains the same connection ID that was requested.

Since clients are always aware whether or not they are using a QUIC proxy, clients are expected to cooperate with proxies in selecting Client Connection IDs. A proxy detects a conflict when it is not able to create a unique mapping using the Client Connection ID (Section 2.4). It can reject requests that would cause a conflict and indicate this to the client by replying with a CLOSE_CLIENT_CID capsule. In order to avoid conflicts, clients SHOULD select Client Connection IDs of at least 8 bytes in length with unpredictable values. A client also SHOULD NOT select a Client Connection ID that matches the ID used for the QUIC connection to the proxy, as this inherently creates a conflict.

If the rejection indicated a conflict due to the Client Connection ID, the client MUST select a new Connection ID before sending a new request, and generate a new packet. For example, if a client is sending a QUIC Initial packet and chooses a Connection ID that conflicts with an existing mapping to the same target server, it will need to generate a new QUIC Initial.

4.2. Adding New Client Connection IDs

Since QUIC connection IDs are chosen by the receiver, an endpoint needs to communicate its chosen connection IDs to its peer before

the peer can start using them. In QUICv1, this is performed using the NEW_CONNECTION_ID frame.

Prior to informing the target of a new chosen client connection ID, the client MUST send a REGISTER_CLIENT_CID capsule request containing the new Client Connection ID.

The client should only inform the target of the new Client Connection ID once an ACK_CLIENT_CID capsule is received that contains the echoed connection ID.

4.3. Sending With Forwarded Mode

Support for forwarding mode is determined by the "Proxy-QUIC-Forwarding" header, see Section 5.

Once the client has learned the target server's Connection ID, such as in the response to a QUIC Initial packet, it can send a REGISTER_TARGET_CID capsule containing the Target Connection ID to request the ability to forward packets.

The client MUST wait for an ACK_TARGET_CID capsule that contains the echoed connection ID before using forwarded mode.

Prior to receiving the proxy server response, the client MUST send short header packets tunnelled in HTTP Datagram frames. The client MAY also choose to tunnel some short header packets even after receiving the successful response.

If the Target Connection ID registration is rejected, for example with a CLOSE_TARGET_CID capsule, it MUST NOT forward packets to the requested Target Connection ID, but only use tunnelled mode. The request might also be rejected if the proxy does not support forwarded mode or has it disabled by policy.

QUIC long header packets MUST NOT be forwarded. These packets can only be tunnelled within HTTP Datagram frames to avoid exposing unnecessary connection metadata.

When forwarding, the client sends a QUIC packet with the Virtual Target Connection ID in the QUIC short header, using the same socket between client and proxy that was used for the main QUIC connection between client and proxy.

If the Virtual Target Connection ID is smaller than the Target Connection ID, the client MUST only write the Virtual Target Connection ID bytes over the start of the Target Connection ID, leaving the remainder of the Target Connection ID unmodified.

```
| 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |

  aa   bb   cc   aa   bb   cc   aa   bb    <--- Target Connection ID
  11   22   33                             <--- Virtual Target Connection
  11   22   33   aa   bb   cc   aa   bb    <--- Resulting Connection ID B
```

If the Virtual Target Connection ID is larger than the Target Connection ID, the client MUST extend the length of the packet by the difference between the two lengths, to include the entire Virtual Target Connection ID.

Clients supporting forwarding mode MUST be able to handle Virtual Target Connection IDs of different lengths than the corresponding Target Connection IDs.

4.4. Receiving With Forwarded Mode

If the client has indicated support for forwarding with the "Proxy-QUIC-Forwarding" header, the proxy MAY use forwarded mode for any Client Connection ID for which it has a valid mapping.

Once a client has sent "Proxy-QUIC-Forwarding" with a value of "?1", it MUST be prepared to receive forwarded short header packets on the socket between itself and the proxy for any Client Connection ID that it has registered with a REGISTER_CLIENT_CID capsule. The client uses the Destination Connection ID field of the received packet to determine if the packet was originated by the proxy, or merely forwarded from the target.

5. Proxy Response Behavior

Upon receipt of a CONNECT request that includes the "Proxy-QUIC-Forwarding" header, the proxy indicates to the client that it supports QUIC-aware proxying by including a "Proxy-QUIC-Forwarding" header in a successful response. If it supports QUIC packet forwarding, it sets the value to "?1"; otherwise, it sets it to "?0".

Upon receipt of a REGISTER_CLIENT_CID or REGISTER_TARGET_CID capsule, the proxy validates the registration, tries to establish the appropriate mappings as described in Section 2.

The proxy MUST reply to each REGISTER_CLIENT_CID capsule with either an ACK_CLIENT_CID or CLOSE_CLIENT_CID capsule containing the Connection ID that was in the registration capsule.

Similarly, the proxy MUST reply to each REGISTER_TARGET_CID capsule with either an ACK_TARGET_CID or CLOSE_TARGET_CID capsule containing the Connection ID that was in the registration capsule.

The proxy then determines the target-facing socket to associate with the client's request. This will generally involve performing a DNS lookup for the target hostname in the CONNECT request, or finding an existing target-facing socket to the authority. The target-facing socket might already be open due to a previous request from this client, or another. If the socket is not already created, the proxy creates a new one. Proxies can choose to reuse target-facing sockets across multiple UDP proxying requests, or have a unique target-facing socket for every UDP proxying request.

If a proxy reuses target-facing sockets, it SHOULD store which authorities (which could be a domain name or IP address literal) are being accessed over a particular target-facing socket so it can avoid performing a new DNS query and potentially choosing a different target server IP address which could map to a different target server.

Target-facing sockets MUST NOT be reused across QUIC and non-QUIC UDP proxy requests, since it might not be possible to correctly demultiplex or direct the traffic. Any packets received on a target-facing socket used for proxying QUIC that does not correspond to a known Connection ID MUST be dropped.

When the proxy recieves a REGISTER_CLIENT_CID capsule, it is receiving a request to be able to route traffic back to the client using that Connection ID. If the pair of this Client Connection ID and the selected target-facing socket does not create a conflict, the proxy creates the mapping and responds with a ACK_CLIENT_CID capsule. After this point, any packets received by the proxy from the target-facing socket that match the Client Connection ID can to be sent to the client. The proxy MUST use tunnelled mode (HTTP Datagram frames) for any long header packets. The proxy SHOULD forward directly to the client for any matching short header packets if forwarding is supported by the client, but the proxy MAY tunnel these packets in HTTP Datagram frames instead. If the mapping would create a conflict, the proxy responds with a CLOSE_CLIENT_CID capsule.

When the proxy recieves a REGISTER_TARGET_CID capsule, it is receiving a request to allow the client to forward packets to the target. The proxy generates a Virtual Target Connection ID for the client to use when sending packets in forwarding mode. If forwarding mode is not supported, the proxy MUST NOT send a Virtual Target Connection ID by setting the length to zero. If forwarding mode is supported, the proxy MUST use a Virtual Target Connection ID that does not introduce a conflict with any other Connection ID on the client-facing socket. The proxy creates the mapping and responds with an ACK_TARGET_CID capsule. Once the successful response is sent, the proxy will forward any short header packets received on

the client-facing socket that use the Virtual Target Connection ID using the correct target-facing socket after first rewriting the Virtual Target Connection ID to be the correct Target Connection ID.

A proxy that supports forwarding mode but chooses not to support rewriting the Virtual Target Connection ID to the Target Connection ID may opt to simply let them be equal. If the proxy does wish to choose a Virtual Target Connection ID, it MUST be able to replace the Virtual Target Connection ID with the Target Connection ID and correctly handle length differences between the two.

If the proxy does not support forwarded mode, or does not allow forwarded mode for a particular client or authority by policy, it can reject all REGISTER_TARGET_CID requests with CLOSE_TARGET_CID capsule.

The proxy MUST only forward non-tunnelled packets from the client that are QUIC short header packets (based on the Header Form bit) and have mapped Virtual Target Connection IDs. Packets sent by the client that are forwarded SHOULD be considered as activity for restarting QUIC's Idle Timeout [QUIC].

5.1. Removing Mapping State

For any registration capsule for which the proxy has sent an acknowledgement, any mappings last until either endpoint sends a close capsule or the either side of the HTTP stream closes.

A client that no longer wants a given Connection ID to be forwarded by the proxy sends a CLOSE_CLIENT_CID or CLOSE_TARGET_CID capsule.

If a client's connection to the proxy is terminated for any reason, all mappings associated with all requests are removed.

A proxy can close its target-facing socket once all UDP proxying requests mapped to that socket have been removed.

5.2. Handling Connection Migration

If a proxy supports QUIC connection migration, it needs to ensure that a migration event does not end up sending too many tunnelled or proxied packets on a new path prior to path validation.

Specifically, the proxy MUST limit the number of packets that it will proxy to an unvalidated client address to the size of an initial congestion window. Proxies additionally SHOULD pace the rate at which packets are sent over a new path to avoid creating unintentional congestion on the new path.

6. Example

Consider a client that is establishing a new QUIC connection through the proxy. It has selected a Client Connection ID of 0x31323334. In order to inform a proxy of the new QUIC Client Connection ID, the client also sends a REGISTER_CLIENT_CID capsule.

The client will also send the initial QUIC packet with the Long Header form in an HTTP datagram.

```
Client                                              Server

STREAM(44): HEADERS             -------->
  :method = CONNECT
  :protocol = connect-udp
  :scheme = https
  :path = /target.example.com/443/
  :authority = proxy.example.org
  proxy-quic-forwarding = ?1
  capsule-protocol = ?1

STREAM(44): DATA                -------->
  Capsule Type = REGISTER_CLIENT_CID
  Connection ID = 0x31323334

DATAGRAM                        -------->
  Quarter Stream ID = 11
  Context ID = 0
  Payload = Encapsulated QUIC initial

           <--------  STREAM(44): HEADERS
                        :status = 200
                        proxy-quic-forwarding = ?1
                        capsule-protocol = ?1

           <--------  STREAM(44): DATA
                        Capsule Type = ACK_CLIENT_CID
                        Connection ID = 0x31323334

/* Wait for target server to respond to UDP packet. */

           <--------  DATAGRAM
                        Quarter Stream ID = 11
                        Context ID = 0
                        Payload = Encapsulated QUIC initial
```

Once the client learns which Connection ID has been selected by the target server, it can send a new request to the proxy to establish a mapping for forwarding. In this case, that ID is 0x61626364. The client sends the following capsule:

```
STREAM(44): DATA                --------->
  Capsule Type = REGISTER_TARGET_CID
  Connection ID = 0x61626364

           <--------  STREAM(44): DATA
                        Capsule Type = ACK_TARGET_CID
                        Connection ID = 0x61626364
                        Virtual Target Connection ID = 0x123412341234
                        Stateless Reset Token = Token
```

Upon receiving an ACK_TARGET_CID capsule, the client starts sending Short Header packets with a Destination Connection ID of 0x123412341234 directly to the proxy (not tunnelled), and these are rewritten by the proxy to have the Destination Connection ID 0x61626364 prior to being forwarded directly to the target. In the reverse direction, Short Header packets from the target with a Destination Connection ID of 0x31323334 are forwarded directly to the client without modification.

7. Packet Size Considerations

Since Initial QUIC packets must be at least 1200 bytes in length, the HTTP Datagram frames that are used for a QUIC-aware proxy MUST be able to carry at least 1200 bytes.

Additionally, clients that connect to a proxy for purpose of proxying QUIC SHOULD start their connection with a larger packet size than 1200 bytes, to account for the overhead of tunnelling an Initial QUIC packet within an HTTP Datagram frame. If the client does not begin with a larger packet size than 1200 bytes, it will need to perform Path MTU (Maximum Transmission Unit) discovery to discover a larger path size prior to sending any tunnelled Initial QUIC packets.

Once a proxied QUIC connections moves into forwarded mode, the client SHOULD initiate Path MTU discovery to increase its end-to-end MTU.

8. Security Considerations

Proxies that support this extension SHOULD provide protections to rate-limit or restrict clients from opening an excessive number of proxied connections, so as to limit abuse or use of proxies to launch Denial-of-Service attacks.

Sending QUIC packets by forwarding through a proxy without tunnelling exposes some QUIC header metadata to onlookers, and can be used to correlate packet flows if an attacker is able to see traffic on both sides of the proxy. Tunnelled packets have similar inference problems. An attacker on both sides of the proxy can use

the size of ingress and egress packets to correlate packets belonging to the same connection. (Absent client-side padding, tunneled packets will typically have a fixed amount of overhead that is removed before their HTTP Datagram contents are written to the target.)

Since proxies that forward QUIC packets do not perform any cryptographic integrity check, it is possible that these packets are either malformed, replays, or otherwise malicious. This may result in proxy targets rate limiting or decreasing the reputation of a given proxy.

9. IANA Considerations

9.1. HTTP Header

This document registers the "Proxy-QUIC-Forwarding" header in the "Permanent Message Header Field Names" <https://www.iana.org/assignments/message-headers>.

Header Field Name	Protocol	Status	Reference
Proxy-QUIC-Forwarding	http	exp	This document

Figure 3: Registered HTTP Header

9.2. Capsule Types

This document registers six new values in the "HTTP Capsule Types" registry established by [HTTP-DGRAM].

Capule Type	Value	Specification
REGISTER_CLIENT_CID	0xffe200	This Document
REGISTER_TARGET_CID	0xffe201	This Document
ACK_CLIENT_CID	0xffe202	This Document
ACK_TARGET_CID	0xffe203	This Document
CLOSE_CLIENT_CID	0xffe204	This Document
CLOSE_TARGET_CID	0xffe205	This Document

Table 1: Registered Capsule Types

10. References

10.1. Normative References

[CONNECT-UDP] Schinazi, D., "Proxying UDP in HTTP", Work in Progress, Internet-Draft, draft-ietf-masque-connect-

udp-15, 17 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-connect-udp-15>.

[HTTP-DGRAM] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", Work in Progress, Internet-Draft, draft-ietf-masque-h3-datagram-11, 17 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-h3-datagram-11>.

[HTTP3] Bishop, M., "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>.

[INVARIANTS] Thomson, M., "Version-Independent Properties of QUIC", RFC 8999, DOI 10.17487/RFC8999, May 2021, <https://www.rfc-editor.org/rfc/rfc8999>.

[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>.

[RFC8941] Nottingham, M. and P-H. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/rfc/rfc8941>.

10.2. Informative References

[QUIC-LB] Duke, M., Banks, N., and C. Huitema, "QUIC-LB: Generating Routable QUIC Connection IDs", Work in Progress, Internet-Draft, draft-ietf-quic-load-balancers-14, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-load-balancers-14>.

Acknowledgments

Thanks to Lucas Pardue, Ryan Hamilton, and Mirja Kuehlewind for their inputs on this document.

Authors' Addresses

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: tpauly@apple.com

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

Email: dschinazi.ietf@gmail.com