Network Working Group
Internet-Draft
Intended status: Informational
Expires: 14 January 2021

A. Chernyakhovsky
D. McCall
D. Schinazi
Google LLC

13 July 2020

Requirements for a MASQUE Protocol to Proxy IP Traffic
draft-cms-masque-ip-proxy-reqs-00

Abstract

There is interest among MASQUE working group participants in designing a protocol that can proxy IP traffic over HTTP. This document describes the set of requirements for such a protocol.

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 14 January 2021.

Copyright Notice

Table of Contents

1. Introduction

There exist several IETF standards for proxying IP in a way that is authenticated and confidential, such as IKEv2/IPsec [IKEV2]. However, those are distinguishable from common Internet traffic and often blocked. Additionally, large server deployments have expressed interest in using a VPN solution that leverages existing security protocols such as QUIC [QUIC] or TLS [TLS] to avoid adding another protocol to their security posture.

This document describes the set of requirements for a protocol that can proxy IP traffic over HTTP. The requirements outlined below are similar to the considerations made in designing the CONNECT-UDP method [CONNECT-UDP], additionally including IP-specific requirements, such as a means of negotiating the routes that should be advertised on either end of the connection.

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

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. Definitions

* Data Transport: The method by which IP packets are transmitted. This can involve streams or datagrams.

* IP Session: An association between client and server whereby both agree to proxy IP traffic given certain configuration properties. This is similar to a Child Security Association in IKEv2 terminology.

2. Use Cases

There are multiple reasons to deploy an IP proxying protocol. This section discusses some examples of use cases that MUST be supported by the protocol.

2.1. Point to Point Connectivity

Point-to-point connectivity creates a private, encrypted and authenticated network between two IP addresses. This is useful, for example, with container networking to provide a virtual (overlay) network with addressing separate from the physical transport. An example of this is Wireguard.

2.2. Point to Network Connectivity

Point-to-Network connectivity is the more traditional remote-access "VPN" use case, frequently used when a user needs to connect to a different network (such as an enterprise network) for access to resources that are not exposed to the public Internet.

2.3. Network to Network Connectivity

Network-to-Network connectivity is also called a site-to-site VPN. Like the point-to-network use case, the goal is to connect to a network that is not exposed publicly. The site-to-site aspects make this transparent to the user; the entire networks are connected to each other and route packets transparently without a VPN client installed on the user's device. This style of connectivity can also be used to connect devices that cannot run VPN clients through to the network.

3. Requirements

This section lists requirements for a protocol that can proxy IP over an HTTP connection.

3.1. IP Session Establishment

The protocol will allow the client to request establishment of an IP Session, along with configuration options and one or more associated Data Transports. The server will have the ability to accept or deny the client's request.

3.2. Proxying of IP packets

The protocol will establish Data Transports, which will be able to forward IP packets, in their unmodified entirety. The protocol will support both IPv6 [IPV6] and IPv4 [IPV4].

3.3. Maximum Transmission Unit

The protocol will allow endpoints to negotiate the Maximum Transmission Unit (MTU) in use over a given Data Transport. This will allow avoiding IP fragmentation, especially as IPv6 does not allow IP fragmentation by nodes along the path.

3.4. IP Assignment

Both the client or server may request to be assigned an IP address range. In response to the request, the peer will respond with an IP address range of its choosing.

3.5. Route Negotiation

At any point in an IP Session (not limited to its initial negotiation), the protocol will allow both client and server to request routes to specific IP address ranges. In response to this request, the peer will have the ability to respond with a subset of routes that it is willing to accept, or deny the request.

3.6. Identity

When negotiating the creation of an IP Session, the protocol will allow both endpoints to exchange an identifier. For example, both endpoints will be able to identify themselves by sending a fully-qualified domain name.

3.7. Transport Security

The protocol MUST be run over a protocol that provides mutual authentication, confidentiality and integrity. Using QUIC or TLS would meet this requirement.

3.8. Authentication

Additionally to the authentication provided by the transport, the protocol will have the ability to authenticate both client and server during the establishment of the IP Session. In particular, it will be possible for the client to offer an OAuth Access Token [OAUTH] to the server when requesting IP proxying, potentially through an extension of the protocol. The protocol will also have the ability to support vendor-specific authentication mechanisms as extensions.

3.9. Reliable Transmission of IP Packets

While it is desirable to transmit IP packets unreliably in most cases, the protocol will provide a mechanism to allow forwarding some packets reliably. For example, when using HTTP/3, this can be accomplished by allowing Data Transports to run over both DATAGRAM and STREAM frames.

3.10. Flow Control

The protocol will allow the ability to proxy IP packets without flow control, at least when HTTP/3 is in use. QUIC DATAGRAM frames are not flow controlled and would meet this requirement. The document defining the protocol will provide guidance on how best to use flow control to improve IP Session performance.

3.11. Indistinguishability

A passive network observer not participating in the encrypted connection should not be able to distinguish an IP proxying session from regular encrypted HTTP Web traffic.

3.12. Support HTTP/2 and HTTP/3

The IP proxying protocol discussed in this document will run over HTTP. The protocol SHOULD strongly prefer to use HTTP/3 [H3] and SHOULD use the QUIC DATAGRAM frames [DGRAM] when available to improve performance. The protocol SHOULD also support HTTP/2 [H2] as a fallback when UDP is blocked on the network path. Proxying IP over HTTP/2 MAY result in lower performance than over HTTP/3.

3.13. Multiplexing

Since recent HTTP versions support concurrently running multiple requests over the same connection, the protocol SHOULD support multiple independent instances of IP proxying over a given HTTP connection.

3.14. Load balancing

Clients and servers should each be able to instantiate new Data Transports. This facilitates multi-threaded servers being able to handle a higher bandwidth of IP proxied packets.

The IP proxying mechanisms need to support load balancing of the traffic sent across the session, such as to another server. The document defining the new protocol should provide guidance for when additional connections and/or sessions should be opened, as opposed to reusing existing ones.

3.15. Extensibility

The protocol will provide a mechanism by which clients and servers can add extension information to the exchange that establishes the IP session. If the solution uses an HTTP request and response, this could be accomplished using HTTP headers.

Once the session is established, the protocol will provide a mechanism that allows reliably exchanging vendor-specific messages in both directions at any point in the lifetime of the IP Session.

4. Non-requirements

This section discusses topics that are explicitly out of scope for the IP Proxying protocol. These topics MAY be handled by implementers or future extensions.

4.1. Addressing Architecture

This document only describes the requirements for a protocol that allows IP proxying. It does not discuss how the IPs assigned are determined, managed, or translated. While these details are important for producing a functional system, they do not need to be handled by the protocol beyond the ability to convey those assignments.

4.2. Translation

Some servers may wish to perform Network Address Translation (NAT) or any other modification to packets they forward. Doing so is out of scope for the proxying protocol. In particular, the ability to discover the presence of a NAT, negotiate NAT bindings, or check connectivity through a NAT is explicitly out of scope and left to future extensions.

4.3. IP Packet Extraction

How packets are forwarded between the IP proxying connection and the physical network is out of scope. This is deliberately not specified and will be left to individual implementations.

5. Security Considerations

This document only discusses requirements on a protocol that allows IP proxying. That protocol will need to document its security considerations.

6. IANA Considerations

This document requests no actions from IANA.

Acknowledgments

The authors would like to thank participants of the MASQUE working group for their feedback.

References

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-00, 26 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-datagram-00.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>.

[H3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-29, 9 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-http-29.txt>.

[IPV4] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[IPV6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[QUIC] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-29, 9 June 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>.

[TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Informative References

[CONNECT-UDP] Schinazi, D., "The CONNECT-UDP HTTP Method", Work in Progress, Internet-Draft, draft-schinazi-masque-connect-udp-00, 16 April 2020, <http://www.ietf.org/internet-drafts/draft-schinazi-masque-connect-udp-00.txt>.

[IKEV2] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[OAUTH] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

Authors' Addresses

Alex Chernyakhovsky
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043,
United States of America

Email: achernya@google.com

Dallas McCall
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043,
United States of America

Email: dallasmccall@google.com

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

Email: dschinazi.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 18 October 2020

D. Schinazi
Google LLC

16 April 2020

The CONNECT-UDP HTTP Method
draft-schinazi-masque-connect-udp-00

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/DavidSchinazi/masque-drafts.

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/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. 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 multiple proxies are involved in a CONNECT-UDP request, all the HTTP connections along the path need to be using HTTP/3 [H3] or later in order for UDP payloads to be sent over QUIC DATAGRAM frames. Additionally, 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, through one or more proxies, which can then be secured using QUIC or another protocol running over UDP.

A client sending a CONNECT-UDP request MUST send the authority form of request-target (Section 5.3 of [RFC7230]); i.e., the request-target consists of only the host name and port number of the tunnel destination, separated by a colon. For example,

```
CONNECT-UDP server.example.com:443 HTTP/1.1
Host: server.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" and ":path" pseudo-header fields MUST be omitted.

* The ":authority" pseudo-header field contains the host and port to connect to (equivalent to the authority-form of the request-target of CONNECT-UDP 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 can establish a tunnel either by directly opening a UDP socket to the request-target or, if configured to use another proxy, by forwarding the CONNECT-UDP request to the next inbound proxy. Any 2xx (Successful) response indicates that the sender (and all inbound proxies) will switch to tunnel mode immediately after the blank line that concludes the successful response's header section; data received after that blank line is from the server identified by the request-target. Any response other than a successful response indicates that the tunnel has not yet been formed and that the connection remains governed by HTTP.

A tunnel is closed when a tunnel intermediary detects that either side has closed its connection: the intermediary MUST attempt to send any outstanding data that came from the closed side to the other side, close both connections, and then discard any remaining data left undelivered.

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

When the HTTP connection between client and proxy supports HTTP/3 datagrams [H3DGRAM], UDP packets can be encoded using QUIC DATAGRAM frames. This support is ascertained by checking receipt of the H3_DATAGRAM SETTINGS Parameter. Note that when there are multiple proxies involved, this support needs to be ascertained on all the HTTP connections that will carry proxied UDP packets.

If the client supports HTTP/3 datagrams and has received the H3_DATAGRAM SETTINGS Parameter 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 server by using the "Datagram-Flow-Id" header (see Section 5).

If there are multiple proxies involved, proxies along the chain MUST check whether their upstream connection supports HTTP/3 datagrams. If it does not, that proxy MUST remove the "Datagram-Flow-Id" header before forwarding the CONNECT-UDP request.

The proxy that is creating the UDP socket to the destination responds to the CONNECT-UDP request with a 2xx (Successful) response, and MUST echo 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 destination. It then encodes the payload of UDP datagrams into the payload of HTTP/3 datagrams.

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 they arrive before the CONNECT-UDP request.

If HTTP/3 datagrams are not supported, the stream is used to convey UDP payloads, by prefixing them with a 16-bit length.

5. Datagram-Flow-Id Header Definition

"Datagram-Flow-Id" is a Item Structured Header [STRUCT-HDR]. Its value MUST be an Integer. Its ABNF is:

```
Datagram-Flow-Id = sh-integer
```

6. Server Handling

Unlike TCP, UDP is connection-less. The HTTP server 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.

Servers can use connected UDP sockets if their operating system supports them, as that allows the HTTP server to rely on the kernel to only send it UDP packets that match the correct 5-tuple. If the server 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 server.

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.

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. HTTP Header

This document will request IANA to register the "Datagram-Flow-Id" header in the "Permanent Message Header Field Names" registry maintained at <https://www.iana.org/assignments/message-headers>.

Header Field Name	Protocol	Status	Reference
Datagram-Flow-Id	http	exp	This document

9. 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-00, 26 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-datagram-00.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>.

[H3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-27, 21 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-http-27.txt>.

[H3DGRAM] Schinazi, D., "Using QUIC Datagrams with HTTP/3", Work in Progress, Internet-Draft, draft-schinazi-quic-h3-datagram-03, 12 March 2020, <http://www.ietf.org/internet-drafts/draft-schinazi-quic-h3-datagram-03.txt>.

[QUIC] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-27, 21 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-27.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>.

[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-17, 15 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-header-structure-17.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.

Author's Address

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

Email: dschinazi.ietf@gmail.com

MASQUE
Internet-Draft
Intended status: Informational
Expires: 11 January 2021

M. Westerlund
M. Ihlar
Z. Sarker
M. Kuehlewind
Ericsson

10 July 2020

Transport Considerations for IP and UDP Proxying in MASQUE
draft-westerlund-masque-transport-issues-00

Abstract

The HTTP Connect method uses back-to-back TCP connections to and from a proxy. Such a solution takes care of many transport aspects as well as IP Flow related concerns. With UDP and IP proxying on the other hand, there are several per-packet and per-flow aspects that need consideration to preserve the properties of end- to-end IP/UDP flows. The aim of this document is to highlight and provide solutions for these issues related to UDP and IP proxying.

Discussion Venues

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

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

Source for this draft and an issue tracker can be found at https://github.com/gloinul/draft-westerlund-masque-transport-issues (https://github.com/gloinul/draft-westerlund-masque-transport-issues).

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 11 January 2021.

Table of Contents

1. Introduction

This document examines several aspects related to UDP [RFC0768] over IP [RFC0791] [RFC8200] (IP/UDP) flows when they are proxied according to the MASQUE proposal over QUIC and using HTTP CONNECT method for flow establishment (Connect-UDP) [I-D.schinazi-masque-connect-udp]. It also looks at how transport protocols on top of UDP use this information and contrast that with both the HTTP Connect method [RFC7231] using either TCP [RFC0793] or QUIC [I-D.ietf-quic-transport] as well as the methods used by the TURN protocol [RFC8656].

Aspects discussed include ECN [RFC3168], Differentiated Services Field and its codepoint (DSCP) [RFC2474], Fragmentation and MTU, ICMP [RFC0792], IPv6 FLOW ID [RFC8200], IPv6 Extension headers, and IPv4 Options [RFC0791]. This document also discusses the use of the UDP checksum and the UDP Length field usage related to UDP Options [I-D.ietf-quic-transport].

1.1. Definitions

* UDP Flow: A sequence of UDP packets sharing a 5-tuple.

* ECN: Explicit Congestion Notification [RFC3168].

* DSCP: Differentiated Service Code Point [RFC2474].

* Proxy: This document uses proxy as synonym for the MASQUE Server or an HTTP proxy, depending on context.

* Client: The endpoint initiating a MASQUE tunnel and UDP/IP relaying with the proxy.

* Target: A remote endpoint the client wishes to establish bi-directional communication with.

* UDP proxying: A proxy forwarding data to a target over an UDP "connection". Data is decapsulate at the proxy and amended by a UDP and IP header before forwarding to the target. Datagram boundaries need to be preserved or signalled between the client and proxy.

* IP proxying: A proxy forwarding data to a target over an IP "connection". Data is decapsulate at the proxy and amended by a IP header before forwarding to the target. Packet boundaries need to be preserved or signalled between the client and proxy.

Address = IP address + UDP port

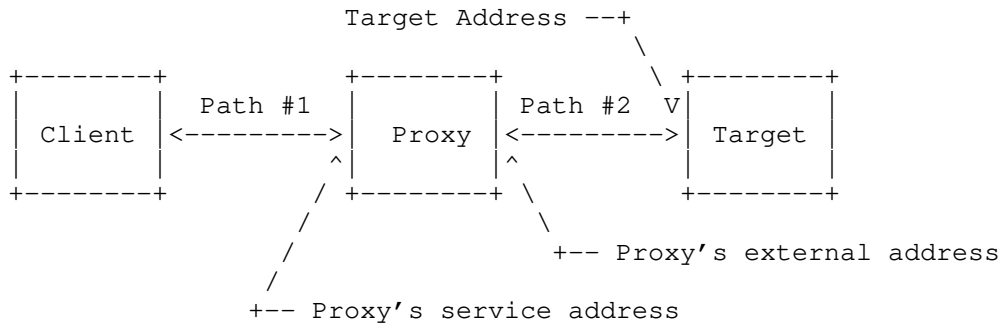

Figure 1: The nodes and their addresses

Figure 1 provides an overview figure of the involved nodes, i.e. Client, Proxy, and Target, that are discussed below. We use the name target for the node or endpoint the client intends to communicate with via the proxy. There are also two network paths. Path #1 is the client to proxy path, where the MASQUE protocol will be used over an HTTP/3 session, usually over QUIC, to tunnel IP/UDP flow(s). Path #2 is the path between the Proxy and the Target.

The client will use the proxy's service address to establish a transport connection on which to communicate with the proxy using the MASQUE protocol. The MASQUE protocol will be used to establish the relaying of a IP/UDP flow from the client using as the source address the proxy's external address and sending to the target address. In addition, after establishment, the reverse is also configured on the proxy; IP/UDP packets sent from the target address to the proxy's external address will be relayed to the client.

1.2. HTTP Connect

The HTTP Connect method [RFC7231] is defined such that the HTTP proxy that receives the request will set up a TCP connection towards a provided (or resolved) target address. After the TCP connection has been established, the proxy will connect the byte stream from the client to the byte stream of the new TCP connection. On byte stream level this is only tunneling. On the transport level on the other hand, two distinct transport connections are established. If the client to proxy connection is HTTP/3, i.e. over QUIC, the basic HTTP Connect method will still lead to TCP connection establishment from proxy to the target servers. In this case the client to proxy QUIC stream's byte stream is connected to the proxy to server TCP connection. For simplicity and clarify the rest of the document will use back to back TCP sessions as a comparison.

Due to the byte stream semantics and the use of transport protocol proxying, most of the transport implications of the header fields and their values are handled on a per path basis, e.g. response to ECN Congestion Experienced (CE) marks. Some information that may be end-to-end related, such as DSCP values, can be copied or translated between the connections if supported by the proxy. MTU is mostly irrelevant and handled fully due to the byte stream nature of the data flowing in the connection. If the MTU differs between the two paths the number of packets required to send a particular data object may differ as well. However, the impact of that is small and depends on the amount of buffering between the two connections. There is no requirement to have a one-to-one correspondence of packets between the two TCP connections.

1.3. TURN

"Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)" [RFC8656] is a solution for relaying UDP datagrams. However, it is a hybrid between a purely encapsulating tunnel and proxying. A somewhat simplified description of the protocol follows below.

A client makes a TURN request over a TCP/TLS connection to a TURN server to authenticate itself and acquire a long-term secret. Note that subsequent requests are secured using keying material from the long term secret exchange. Next, the client can send a request over UDP to be allocated a UDP port number on one of the server's external IP addresses. After that, the client knows the IP address and UDP port number that will be used for the TURN server's side of any UDP flow sent to or received from the remote target.

A TURN client can send UDP packets in two ways. The first solution is to include a send indication for each UDP packet to be relayed. The send indications contains the target destination address and port. The other solution is to create a binding, where a channel ID between the client and TURN server is bound to a specific 5-tuple from the TURN Server to the target destination. The established channel is bi-directional. When a channel has been established, UDP packets can be relayed with a low overhead of 4 bytes.

To receive UDP packets on the allocated port, the client must specify what source address and port (target address) is to be allowed, this is called establishing a permission. Binding a channel creates a permission at the same time. When the TURN server receives a UDP packet from an external source where permission exists but no channel, it will relay the data using a DATA indication, which includes the source address.

A relevant aspect for the rest of the discussion in this document is that TURN has a one to one mapping between IP/UDP/TURN messages between client and TURN server and the IP/UDP packet received or sent on the external side. Also, though TURN messages and indications include header information in the UDP payload, there are distinct IP/UDP headers on each side of the TURN server. Because of this the TURN protocol is able to preserve the per flow and per packet functionalities that exist in IP/UDP headers. For instance, ECN and DSCP markings associated with specific packets are preserved by the relay by copying or translating them from incoming packets to outgoing packets.

1.4. HTTP Connect-UDP

The MASQUE WG is chartered to work on the tunneling of UDP datagrams in a QUIC connection between client and a MASQUE server (We will refer to the MASQUE server as a proxy in the rest of this document to align with the basic HTTP Connect terminology). The proxy forwards tunneled UDP payload to the correct target address. An HTTP Connect-UDP request is used to establish the tunnel and provide the required addressing information.

In the review performed in this document we make the assumption that it is desirable to minimize the per-packet overhead. Specifically, we assume that IP/UDP header information from packets exchanged between the proxy and target will not be sent within the tunnel. The initial relay exchange establishment needs to be performed for each target the client wants to communicate with, i.e. one relay exchange per IP/UDP flow on the proxy to target path. Keeping this state in the proxy on a per UDP flow basis appears trivial. Therefore, the

review will determine what IP/UDP state is required per flow and what is per per packet information.

In this review we also aim to identify the impact on the end-to-end flows by using QUIC as a tunneling protocol, both in stream and/or datagram mode. Properties of IP/UDP flows and higher level transport protocols such as QUIC or RTP will be considered.

Two high level observations need to be made when comparing Connect-UDP to TURN. The first is that QUIC is a proper transport protocol with congestion control. This means that there might not be a one-to-one mapping between events that impact the transport, such as congestion indications, on either path relative to the proxy. The second observation is that tunneled UDP datagrams can be coalesced in a single UDP datagram with either multiple QUIC packets, or multiple QUIC datagram frames in a single QUIC packet. If reliable streams are used instead of datagram frames, then a UDP payload may even be fragmented over multiple UDP packets containing QUIC packets.

This all motivates a very careful consideration of the IP and UDP header information and how that needs to be handled on flow level or per packet level to avoid breaking end-to-end transport properties for the flow.

2. Review of IP Header

2.1. Base Header Fields

This section reviews the header fields in IPv4 [RFC0791] and IPv6 [RFC8200]. It will note which field are version specific. Size differences are not considered in the review as it is focused on functionality and impact.

2.1.1. Version

The proxy needs to know which IP version to use on the proxy to target path. If an explicit IP address is included in the Connect-UDP request, the proxy would know this directly from the IP address format. In the case where a domain name is used to obtain the target IP address, the IP version needs to be specified or be based on preferences when resolving the domain name.

It should be noted that different IP versions may be used on the two paths requiring the proxy to do translation. This can also lead to scenarios whereby version specific information carried on one path does not translate to the version used on the other path.

2.1.2. DSCP

Diffserv code points are primarily used to indicate forwarding behavior. Codepoints on IP/UDP flows on the proxy to target path are either set on a flow level or packet level. Codepoints set on a flow level are set at flow instantiation and can be updated during ongoing relaying.

A DSCP value received on IP/UDP packets in the target to proxy direction may be propagated to the Proxy to client path. However, that is problematic when the datagram is tunneled in QUIC. The DART considerations [RFC7657] for connection oriented transport applies here. If multiple DSCPs are used for a single connection, then there would be a need for having separate congestion control states for the different forwarding behaviors, which would likely require QUIC protocol extensions. The same issue exists for packets sent by the client on the client to proxy path.

Different forwarding behaviors in both directions of the path connecting the client and the proxy could be enabled without a QUIC extension by establishing individual QUIC connections per forwarding behavior used. However, this requires that the proxy is able to bind multiple QUIC connections received from the client into a single IP/UDP flow on the proxy to target path. This could have significant security model implications as authorization would be needed to add subsequent bindings to an existing flow.

Let's consider the capability for the proxy to send packets with packet-level DSCP marks towards the target. That would require at a minimum a per packet indication mechanism and would enable different forwarding behaviors on the proxy to target path. Similarly, a per packet mechanism would be needed for the proxy to be able to relay DSCP values received from the target towards the client. The usefulness of the latter would be to ensure that the transport protocol on top of MASQUE is able to determine whether there is a need for multiple congestion control states for different sub-sets of packets within the received IP/UDP flow. WebRTC IP/UDP flows could have this property.

The most basic DSCP relay capability would be to set the same DSCP value on all IP/UDP packets sent by the proxy to the same target for a specific IP/UDP flow. This capability would only require signalling of the desired value at flow establishment. A mechanism to update the DSCP value for an ongoing flow should also be considered.

Another issue with DSCP mapping to forwarding behaviors is that the mappings are defined per network location, typically within one

administrative domain of routing. Therefore they may be remapped on the different paths relative to the proxy. When the client and proxy reside in two different administrative domains there will be an additional challenge for the client to use the right DSCP value. Thus, support for DSCP in the MASQUE protocol should either be limited to consider per hop behavior or the use of a mapping table such that the proxy can translate an incoming DSCP value to a locally used value.

2.1.3. ECN

The Explicit Congestion Notification (ECN) [RFC3168] field carries per packet path signals about congestion. The discussion of ECN capability can be split into two parts, one for each path relative to the proxy. On the client to proxy path the QUIC connection used for tunneling the UDP datagrams can enable and use ECN on that path specifically. Any congestion experienced (ECN-CE) marking on that path impacts the congestion window of the client to proxy QUIC connection, thus indirectly affecting the end-to-end flow.

The capability to use ECN on the proxy to target path requires proxy protocol support. This will enable the end-to-end usage of ECN in the upper layer transport protocol. To support ECN end-to-end when using MASQUE proxy two functionalities need to exist.

First, the capability of setting the ECN field value (Not-ECT, ECT(1), ECT(0)) on any IP/UDP packets sent from proxy to target. This value can be set initially but may be changed during ongoing IP/UDP flow proxying, as the end-to-end transport may subsequently determine that the path is not ECN capable.

Secondly, the client must be able to receive per packet indications of the ECN field value for every packet received by the proxy from the target. This ensures that the upper layer transport protocol receives ECN information per relayed UDP datagram. The information will be used to react to ECN-CE (Congestion Experienced) marks and for validation of the ECN path capability.

A solution like TURN's [RFC8656] translation of ECN markings between the two paths is not possible for multiple reasons. First, ECN marks on the client to proxy path will be consumed and reacted to by the QUIC connection used for tunneling. Second, the previously discussed lack of a one-to-one relationship of IP/UDP packets prevents accurate tracking of the ECN markings and will make the end-to-end validation fail. Therefore additional explicit signaling between the proxy and the client would be needed.

2.1.4. Identification, Flags, and Fragmentation offset (IPv4 Only)

These fields are used for the IPv4 fragmentation solution. The authors are of the opinion that IP level fragmentation should be avoided. However, since there are no guarantees for a one-to-one packet relation between the two paths relative to the proxy, any IPv4 fragments will need re-assembly upon reception by the endpoints and the proxy.

To support Path MTU Discovery the Don't Fragment (DF) bit needs to be set for all outgoing IP/UDP packets from the proxy to the target. Per flow or per packet setting of this bit needs to be supported.

2.1.5. Flow Label (IPv6 Only)

The IPv6 flow label is used by the network to identify flows, for example to prevent a single flow to be spread over multiple paths when load balancing based on Equal Cost Multipath (ECMP) routing [RFC6438] is performed. The flow label should be set by the endpoint originating the IP/UDP flow, as it knows when a flow qualifies for a unique IPv6 flow label. Thus, it is expected that one IPv6 flow label will be used for the IP/UDP flow that carries the client to proxy QUIC connection, and one for each IP/UDP flow established by the MASQUE protocol to different target addresses.

Based on the above reasoning it does not seem like there is a need for the MASQUE protocol to explicitly signal or indicate flow labels.

2.1.6. Total Length / Payload length

The Total Length (IPv4) / Payload length (IPv6) fields contain the size of the IP packet, either directly for IPv4, or indirectly in IPv6 (by providing the length after the fixed 40-byte header, i.e. for extension headers and data).

These field are necessary for the processing on reception, however it does not need to be communicated on a per packet-basis to the client, or be provided by the client, with a single potential exception that is discussed in the context of the UDP length field (Section 3.2).

2.1.7. Protocol / Next Header

The Protocol (IPv4) and Next Header (IPv6) fields provide the identification of the upper layer protocol, in this case UDP. For IPv6 one or more extension headers may first be identified in a chain before arriving at UDP.

The use of UDP relaying will need to be signalled explicitly to separate it from other types of relaying, such as the IP tunneling/ relaying discussed in the MASQUE charter.

2.1.8. Time to Live / Hop Limit

The purpose of the Time to Live (IPv4) and Hop Limit (IPv6) fields is to prevent packets from having an infinite life time in case of routing loops. The acronym TTL is used from here on to describe any of these fields. TTL limits the number of routing hops a packet survives and should result in an ICMP message back to the sender when it expires. Therefore, it is possible to use TTL for investigating network paths.

It is not clear if such a mechanism needs to be supported in a MASQUE protocol context. If something like trace-route is to be supported, per packet setting of the TTL field would be needed.

The need for echoing the TTL field value on reception of a IP/UDP packet from the target to the client appears also very limited. The value set on transmission of a packet is usually an operating system set default value.

The authors believe that the proxy's default values are sufficient for the MASQUE protocol functionality.

2.1.9. Header Checksum (IPv4 Only)

The IPv4 checksum field verifies the integrity against non-intentional errors in transmission or processing of the IP header. IPv6 lacks this checksum and instead relies on the transport protocol checksum.

The value is generated when an IP packet is transmitted from the proxy and verified on reception. No further functionality required.

2.1.10. Source and Destination Address

On the path from the proxy to the target, the source address will be the proxy's external address applied when relaying the IP/UDP packets. This address will be determined as part of the IP/UDP flow tunneling establishment and should be signalled back to the client by the proxy. The destination address used will be the target's IP address.

The source addresses used on the client to proxy path are only needed for the communication between the client and proxy and are part of the QUIC connection's state. Thus, the possibility to change it will

depend on the mechanisms in QUIC for dealing with client address migration or multi-path. Further discussion should not be needed.

In case of IP proxying, as the proxy cannot utilize port numbers, the proxy might need to maintain multiple external IP addresses in order to identify different forwarding processes for packet received from multiple target servers. If the proxy is guaranteed to be on-path between the client and server the proxy could also conserve the client's source IP address as it's external address.

The source and destination addresses are therefore part of the fundamental state for IP/UDP flow relaying and need to be established at initiation. The 2 (IP proxying) or 5-tuple (IP/UDP proxying) from the proxy to the target and the reverse tuple needs to be explicitly signalled. The client either needs to explicitly provide the target IP address or a domain name that the proxy can resolve to a target IP address. The proxy needs to notify the client about which source IP address it uses when sending on the proxy to target path.

2.2. IPv4 Options Header

The use of IPv4 Options header on the general Internet is very limited. It is therefore likely that no functionality is required.

2.3. IPv6 Extension Headers

One IPv6 Extension header that needs discussion here is the fragmentation header. Although it is the IP originating node that adds the fragmentation header, the MASQUE protocol will likely need to control whether IPv6 fragmentation should be used or not, in the same way as for the IPv4 DF bit.

Some existing IPv6 extension headers could be added by the originating node. Whether they require any explicit signalling or relaying of data to the client needs to be investigated further. Especially Hop-by-Hop options, such as the IPv6 Minimum Path MTU Hop-by-Hop Option [I-D.ietf-6man-mtu-option].

There are also some individual proposals for extension header that might matter in the future: Network Tokens [I-D.yiakoumis-network-tokens], IPv6 Truncate Option [I-D.leddy-6man-truncate]. Thus, consideration needs to be made if there are necessary to future proof the Masque protocol, at least to enable future extensions to support per packet Extension headers.

3. Review of UDP Header

3.1. Source and Destination Port

For UDP proxying, the UDP destination port is used by endpoints to locate the destination process that should consume a specific UDP datagram. Source Ports can be used by the receiving application to separate flows based on the 5-tuple.

As discussed in Section 2.1.10 the UDP source and destination ports are part of the 5-tuple and needs to be communicated on IP/UDP flow establishment.

3.2. UDP Length

The UDP length field specifies the UDP payload length in octets.

The UDP length field normally indicates that the UDP payload fills up the remainder of the IP packet. However, this is not always the case. Specifically, UDP options [I-D.ietf-tsvwg-udp-options] are designed to make use of the surplus area between the end of the UDP data section and the end of the IP packet.

Thus, for the MASQUE protocol to preserve the capability to carry UDP options in UDP relaying this surplus area and the UDP payload data length field need to transmitted from client to proxy in both directions.

3.3. UDP Checksum

The UDP checksum verifies the UDP datagram headers and payload and the pseudo header with IP layer information. The UDP checksum should always be verified by the receiving party. The UDP checksum may also be set to zero to provide no verification. This is primarily used by tunnel encapsulation formats. If it is desired to send IP/UDP flows from the proxy to the target address with a zero UDP checksum, the MASQUE protocol needs to support an indication of this desire.

The use of zero UDP checksum for IPv6 is more restricted than for IPv4 due to the lack of a IP header checksum [RFC6936].

The authors do not believe it is necessary to be able to send IP/UDP datagrams with a zero UDP checksum. It is also not necessary to relay the UDP checksum generated by the target, since the QUIC protocol will provide stronger cryptographic integrity verification of the UDP datagram payload. Also, for the target generated checksum to be meaningful to the client the complete datagram and pseudo header would need to be reconstructed, which in would likely require

extra processing and copying of data. Though some cases of erroneous handling of IP/UDP header fields by the MASQUE protocol could be detected in this way, it is not deemed to be worth the effort.

For the packets the client sends to be relayed to the target destination, having the client create a UDP checksum would provide some protection against processing or implementation errors, although the overhead and extra processing is likely not worth the effort.

In addition the calculation and verification of the transport header checksum is one of these aspects that can be offloaded to hardware in a proxy.

4. ICMP

Internet Control Message Protocol (ICMP) [RFC0792][RFC4443] messages are useful hints on why sent IP packets appear to disappear in the network. Especially for what might appear to be intermittent issues, such as exceeding the MTU of the path.

Lets start with clarifying which ICMP messages may be relevant to handle in the MASQUE protocol. The primary concern is to ensure that the client receives any ICMP responses that are sent back to the proxy as result of packets the client had relayed through the proxy towards the target destination. The proxy should validate and identify ICMP messages that relate to a particular IP/UDP flow that the proxy sends. The relevant ICMP information, i.e. Type and Code as well as any included bytes from the packets that can be used to identify the actual IP/UDP packet in the client, should be sent to the client. Such a functionality would enable the the client to receive Packet Too Big messages that speeds up Path MTU Discovery (Section 5). It also enables the client to learn when there appears to be no one at the target address, i.e. the port, host or network unreachable codes.

IP/UDP packets reaching the proxy from a target address may result in that ICMP messages are sent back to the target. For example a port unreachable message would be sent if a packet arrives after the client has terminated the tunneling session.

Any ICMP packets that result from IP/UDP packets exchanged between the client and the proxy, related to the QUIC connection is to be validated and consumed by the QUIC implementation.

Thus, depending on the ICMP message, the MASQUE protocol needs to consider a mechanism for the proxy to indicate to the client that it has received and validated ICMP messages. If the ICMP message indicates a connection failure, HTTP response error codes can be

used. However, for HTTP Connect-UDP the response code was sent when the UDP socket was created, while an ICMP message would only be received after UDP packets have been relayed.

5. Maximum Transmission Unit (MTU)

The MTU available for the UDP payload depends on whether the QUIC connection uses datagrams or streams to carry the UDP payload on the client to proxy path. When streams are used, the outer QUIC connection can fragment and re-assembly UDP Payloads of any size. In this case any MTU issues will arise on the proxy to target path.

When using datagrams, unless the QUIC datagram extension provides a fragmentation solution, then the outer QUIC connection will provide a MTU that is dependent on the largest datagram payload that can be transmitted. A potential issue is that this might be variable over time, both due to underlying path changes, and to variable elements of the QUIC protocol. However, a base overhead from the QUIC headers should be possible to calculate based on the maximum QUIC packet size. Depending on the amount of per packet information needed to be provided additional headroom for the MASQUE encapsulation may be required.

To enable PMTUD discovery certain aspects are needed or greatly simplify the process.

* Ensure that the DF bit is set to Don't Fragment on outgoing IPv4/ UDP packets from the proxy to the target. For IPv6 the use of the fragmentation header needs to be prevented.

* Have the proxy signal back to the client its current interface MTU limit for packets that will be sent from proxy to target.

* Have the tunneling QUIC connection expose the current MTU for datagrams to the MASQUE implementation. This is likely dynamic and can be updated at any point.

* Returning ICMP Packet Too Big (PTB) message from the proxy to the client when packets are dropped due to MTU on the proxy to target path.

It should be noted that unless the QUIC datagram extension provides a fragmentation mechanism this will in many scenarios be the most likely MTU bottleneck and there is no work around for it, the IP/UDP tunneled traffic will need to fit or be dropped.

5.1. IPv6 Fragmentation

Reassembly of received traffic will occur on each node, Client, Proxy, and Target. The need for IP level fragmentation should be avoided, by having the working MTU be propagated up. Initial MTU signaling should exist for the flow. However, this is potentially dynamic and a PMTUD process running for the outer QUIC tunnel on the client to proxy path will update the supported MTU.

An option for controlling if fragmentation should occur or not by the Proxy should be considered.

5.2. IPv4 Fragmentation

As IPv4 fragmentation is flexible and allows an on-path node to fragment a packet, enabling fragmentation may reduce the MTU issues. However, Path MTU Discovery is recommended instead, for the following reasons:

* Fragmentation increases the packet loss rate.

* IP fragments do not traverse NATs and Firewalls well. Which is especially relevant for MASQUE as significant deployments will be clients in access or residential networks that have NATs or Firewalls on the path from the client to the proxy.

6. Summary

Lets sum up the aspects of the IP/UDP header fields and related protocols in a couple of categorizes.

6.1. UDP Flow Information and configuration

This section contains header fields whose value either will be static for a given IP/UDP flow, apply until changes, or can be used as default values when per packet values are not given.

First of all, the IP source and destination address as well as UDP source and port information is directly related to the establishment of a bi-directional IP/UDP flow between proxy and the target. The desired IP version also needs to be indicated if the address is expressed as hostname, but otherwise would be given by the format of the address. The target always needs to be provided by the client. Since there are cases where the client would need to know the external address used by the proxy towards the target, there needs to be a way for the client to request this information.

The upper layer protocols between the client and the target might have different capabilities of using ECN. Therefore it is necessary to be able to signal whether the ECN fields in proxy to target packets should be set to Not-ECT, ECT(0) or ECT(1).

If a DSCP marking other than 0, i.e. best effort, is desired then a default value could be set as part of the relay establishment. This value could potentially be overridden on a per packet basis or be changed at a future point in the relayed flows lifetime.

A don't fragment setting could likely be defaulted to be always true. However, we invite further discussion if there are cases where it would be better to enable IP level fragmentation for some packets, or for all.

The Hop Limit could likely be using node default values, unless someone raises a use case where they actually want to modulate the value on per packet basis or set an explicit value for the IP/UDP flow.

The MTU known by the proxy towards the indicated target should be signalled back at relay establishment by the proxy.

6.2. Potential Per Packet Information

This section contains information that would be necessary to associate with a specific packet when the client request sending or the proxy relays a received packet.

For each packet the proxy receives the ECN field's value need to be relayed to the client so that the upper layer can respond to either a CE marking indicating congestion, or any remarking.

There are examples where IP/UDP flows contain packets that do not have uniform DSCP marking. To enable sending of such streams, any DSCP value other the default value would need to be indicated by the client to the proxy.

If it is desired to verify the UDP Checksum in the client to avoid any potential errors the MASQUE protocol implementation may cause the received UDP checksum value would need to be relayed to the client. The UDP checksum value could also be calculated by the client and included.

To support UDP Options, the UDP option and their values needs to be signalled.

If support for including IPv6 Extension Headers that needs client side information then the extension header information would need to be indicated by the client and also tunnelled.

6.3. Event based Interactions

This section summarizes information that are triggered by events and not directly connected to a specific packet in the IP/UDP flow being relayed. Instead these are more of the nature of asynchronous events caused by the network, the upper layer transport protocol, or application.

The ECN setting for packets sent by the proxy to the target may need to change. If the upper layer transport protocol determines that the end-to-end path is not ECN capable it will need to change an ECT marking to Not-ECT. Some protocols may defer probing for ECT capability until after some initial handshake and packet exchange has occurred.

The upper layer application may change its desired forwarding behavior for the packets in the IP/UDP stream, thus the need for the client to change the default applied DSCP value on packets sent by the proxy to target.

The reception of ICMP messages by the proxy will likely be the result of a packet sent to the target but the cause is likely in the network. When this occurs the client needs to be informed of this ICMP message, especially when this event leads to a connection failure.

The proxy may detect an MTU change on the proxy to target path due to received ICMP messages that are consumed in the IP stack of the proxy and affecting the MTU for outgoing packets. Thus, the proxy may need to indicate to the client that it no longer can send non-fragmented packets larger than the new MTU.

7. Conclusion

This document shows that many of the fields in the IP/UDP header can be signaled initially at flow establishment. Especially IP addresses and potentially ports need to be communicated as those also need to be used for flow identification. It also shows that certain field values or related information needs to be relayed or signalled based on asynchronous application or network events. Other field information could need per packet relaying. The latter requires a more active bidirectional communication channel between the proxy and the client.

These aspects need to be taken into account in the future protocol development of the CONNECT-UDP method to ensure that the MASQUE protocol doesn't prevent future IP and transport protocol evolution.

8. References

8.1. Normative References

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

[I-D.ietf-tsvwg-udp-options]
Touch, J., "Transport Options for UDP", Internet-Draft, draft-ietf-tsvwg-udp-options-08, 12 September 2019, <http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-udp-options-08.txt>.

[I-D.schinazi-masque-connect-udp]
Schinazi, D., "The CONNECT-UDP HTTP Method", Internet-Draft, draft-schinazi-masque-connect-udp-00, 16 April 2020, <http://www.ietf.org/internet-drafts/draft-schinazi-masque-connect-udp-00.txt>.

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

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP",

RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

8.2. Informative References

[I-D.ietf-6man-mtu-option] Hinden, R. and G. Fairhurst, "IPv6 Minimum Path MTU Hop-by-Hop Option", Internet-Draft, draft-ietf-6man-mtu-option-02, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-6man-mtu-option-02.txt>.

[I-D.leddy-6man-truncate] Leddy, J., Bonica, R., and I. Lubashev, "IPv6 Packet Truncation", Internet-Draft, draft-leddy-6man-truncate-05, 10 October 2018, <http://www.ietf.org/internet-drafts/draft-leddy-6man-truncate-05.txt>.

[I-D.yiakoumis-network-tokens] Yiakoumis, Y., McKeown, N., and F. Sorensen, "Network Tokens", Internet-Draft, draft-yiakoumis-network-tokens-01, 18 June 2020, <http://www.ietf.org/internet-drafts/draft-yiakoumis-network-tokens-01.txt>.

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

[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/info/rfc4443>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums",

RFC 6936, DOI 10.17487/RFC6936, April 2013, <https://www.rfc-editor.org/info/rfc6936>.

[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>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <https://www.rfc-editor.org/info/rfc7657>.

[RFC8656] Reddy, T., Ed., Johnston, A., Ed., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 8656, DOI 10.17487/RFC8656, February 2020, <https://www.rfc-editor.org/info/rfc8656>.

Authors' Addresses

Magnus Westerlund
Ericsson

Email: magnus.westerlund@ericsson.com

Marcus Ihlar
Ericsson

Email: marcus.ihlar@ericsson.com

Zaheduzzaman Sarker
Ericsson

Email: zaheduzzaman.sarker@ericsson.com

Mirja Kuehlewind
Ericsson

Email: mirja.kuehlewind@ericsson.com