QUIC Working Group
Internet-Draft
Intended status: Experimental
Expires: May 7, 2020

M. Piraux, Ed.
O. Bonaventure
UCLouvain

November 04, 2019

Tunneling Internet protocols inside QUIC
draft-piraux-quic-tunnel-00

Abstract

This document specifies methods for tunneling Internet protocols such as TCP, UDP, IP and QUIC inside a QUIC connection.

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 May 7, 2020.

Table of Contents

1. Introduction

Mobile devices such as laptops, smartphones or tablets have different requirements than the traditional fixed devices. These mobile devices often change their network attachment. They are often attached to trusted networks, but sometimes they need to be connected to untrusted networks where their communications can be eavesdropped, filtered or modified. In these situations, the classical approach is to rely on VPN protocols such as DTLS, TLS or IPSec. These VPN protocols provide the encryption and authentication functions to protect those mobile clients from malicious behaviors in untrusted networks.

On the other hand, these devices are often multihomed and many expect to be able to perform seamless handovers from one access network to another without breaking the established VPN sessions. In some situations it can also be beneficial to combine two or more access

networks together to increase the available host bandwidth. A protocol such as Multipath TCP supports those handovers and allows aggregating the bandwidth of different access links. It could be combined with single-path VPN protocols to support both seamless handovers and bandwidth aggregation above VPN tunnels. Unfortunately, Multipath TCP is not yet deployed on most Internet servers and thus few applications would benefit from such a use case.

The QUIC protocol opens up a new way to find a clean solution to this problem. First, QUIC includes the same encryption and authentication techniques as deployed VPN protocols. Second, QUIC is intended to be widely used to support web-based services, making it unlikely to be filtered in many networks, in contrast with VPN protocols. Third, the multipath extensions proposed for QUIC enable it to efficiently support both seamless handovers and bandwidth aggregation.

In this document, we explore how (Multipath) QUIC could be used to enable multi-homed mobile devices to communicate securely in untrusted networks. Section 3 describes the reference environment of this document. Then, we explore and compare two different designs. The first, explained in Section 4, uses the recently proposed datagram extension ([I-D.pauly-quic-datagram]) for QUIC to transport plain IP packets over a Multipath QUIC connection. The second, explained in Section 5, uses the QUIC streams to transport TCP bytestreams over a Multipath QUIC connection.

Section 6 specifies how a connection is established in this document proposal. Section 7 specifies the format of the messages introduced by this document. Section 8 contains example flows.

Our starting point for this work is Multipath QUIC that was initially proposed in [CoNEXT]. A detailed specification of Multipath QUIC may be found in [I-D.deconinck-quic-multipath]. Two implementations of different versions of this protocol are available [CoNEXT], [SIGCOMM19].

2. 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.

3. Reference environment

We consider a multihomed client that is attached to one or several access networks. It establishes a Multipath QUIC connection to a concentrator. This MPQUIC connection is used to carry the UDP and TCP packets sent by the client. Thanks to the security mechanisms used by the Multipath QUIC connection, the client data is protected against attacks in one or both of the access networks. The client trusts the concentrator. The concentrator decrypts the QUIC packets exchanged over the Multipath QUIC connection and interacts with the remote hosts as a VPN concentrator would do.

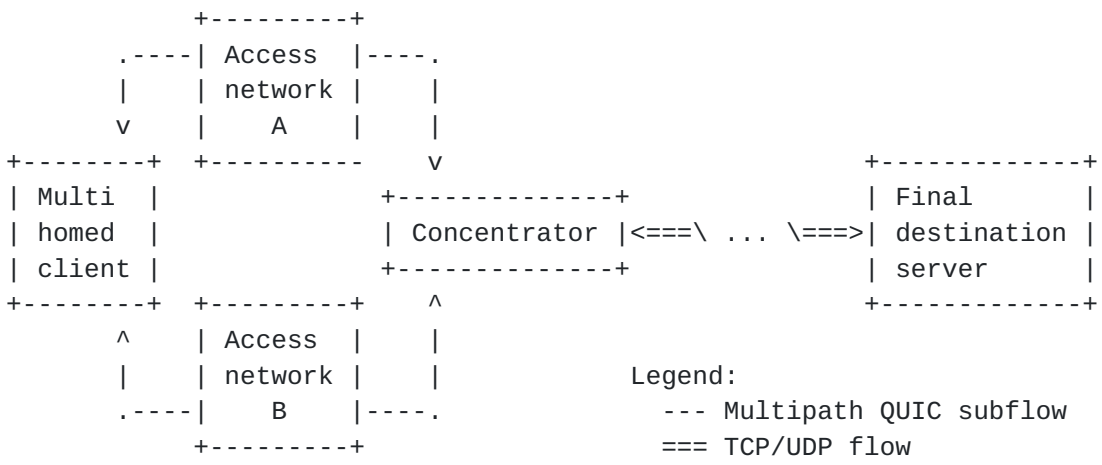


Figure 1: Example environment

Figure 1 illustrates a client-initiated flow. We also discuss inbound connections in this document in Section 6.

4. The datagram mode

Our first mode of operation, called the datagram mode in this document, enables the client and the concentrator to exchange raw IP packets through the Multipath QUIC connection. This is done by using the recently proposed QUIC datagram extension [I-D.pauly-quic-datagram]. In a nutshell, to send an IP packet to a remote host, the client simply passes the entire packet as a datagram to the Multipath QUIC connection established with the concentrator. The IP packet is encoded in a QUIC DATAGRAM frame, then encrypted and authenticated in a QUIC packet. This transmission is subject to congestion control, but the datagram that contains the packet is not retransmitted in case of losses as specified in [I-D.pauly-quic-datagram]. The datagram mode is intended to provide a similar service as the one provided by IPSec tunnels or DTLS.

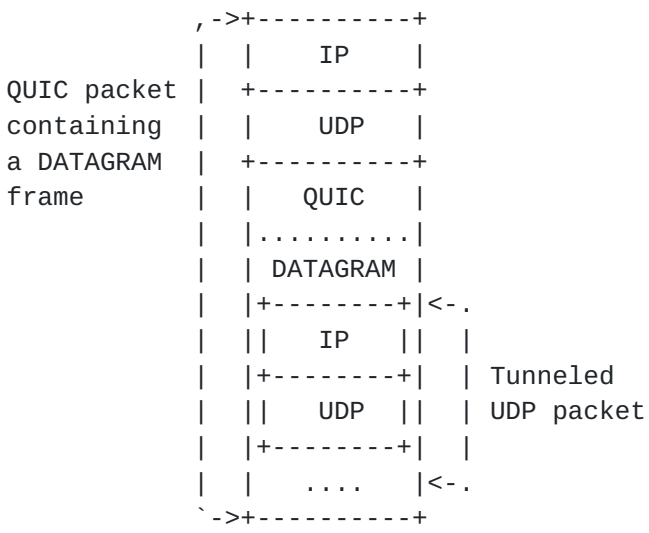


Figure 2: QUIC packet sent by the client when tunneling a UDP packet

Figure 2 illustrates how a UDP packet is tunneled using the datagram mode. The main advantage of the datagram mode is that it supports IP and any protocol above the network layer. Any IP packet can be transported using the datagram extension over a Multipath QUIC connection. However, this advantage comes with a large per-packet overhead since each packet contains both a network and a transport header. All these headers must be transmitted in addition with the IP/UDP/QUIC headers of the Multipath QUIC connection. For TCP connections for instance, the per-packet overhead can be large.

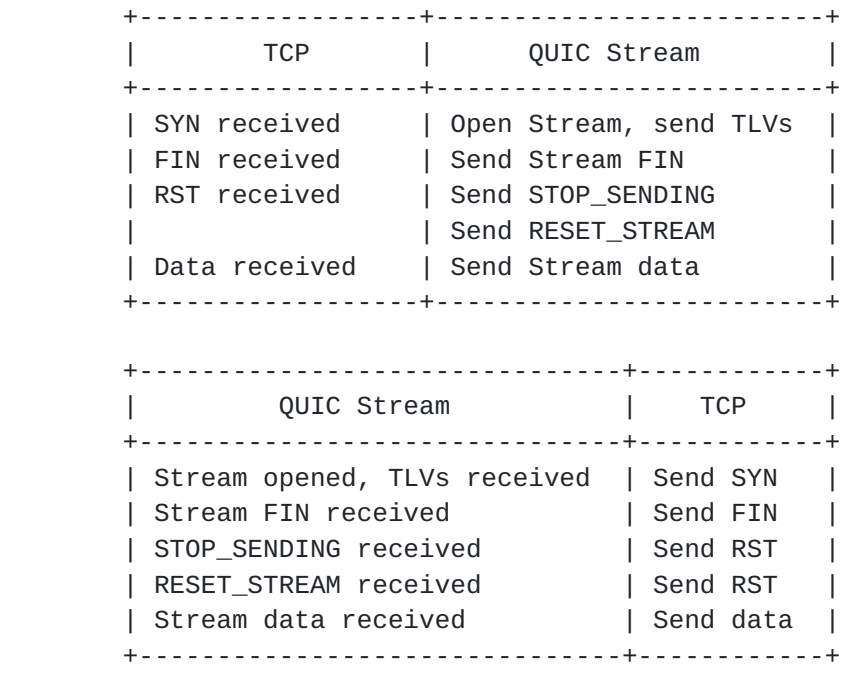
5. The stream mode

Since QUIC support multiple streams, another possibility to carry the data exchanged over TCP connections between the client and the concentrator is to transport the bytestream of each TCP connection as one of the bidirectional streams of the Multipath QUIC connection. For this, we base our approach on the 0-RTT Converter protocol [I-D.ietf-tcpm-converters] that was proposed to ease the deployment of TCP extensions. In a nutshell, it is an application proxy that converts TCP connections, allowing the use of new TCP extensions through an intermediate relay.

We use a similar approach in our stream mode. When a client opens a stream, it sends at the beginning of the bytestream one or more TLV messages indicating the IP address and port number of the remote destination of the bytestream. Their format is detailed in section Section 7.1. Upon reception of such a TLV message, the concentrator opens a TCP connection towards the specified destination and connects the incoming bytestream of the Multipath QUIC connection to the

bytestream of the new TCP connection (and similarly in the opposite direction).

Figure 3 summarizes how the new TCP connection is mapped to the QUIC stream. Actions and events of a TCP connection are translated to action and events of a QUIC stream, so that a state transition of one is translated to the other.



TCP	QUIC Stream
SYN received	Open Stream, send TLVs
FIN received	Send Stream FIN
RST received	Send STOP_SENDING
	Send RESET_STREAM
Data received	Send Stream data

QUIC Stream	TCP
Stream opened, TLVs received	Send SYN
Stream FIN received	Send FIN
STOP_SENDING received	Send RST
RESET_STREAM received	Send RST
Stream data received	Send data

Figure 3: TCP connection to QUIC stream mapping

The QUIC stream-level flow control can be tuned to match the receive window size of the corresponding TCP, so that no excessive data needs to be buffered.

6. Connection establishment

The client MUST establish a connection using the Multipath Extensions defined in [I-D.deconinck-quic-multipath].

During connection establishment, the QUIC tunnel support is indicated by setting the ALPN token "qt" in the TLS handshake. Draft-version implementations MAY specify a particular draft version by suffixing the token, e.g. "qt-00" refers to the first version of this document.

The concentrator can control the number of connections bytestreams that can be opened initially by setting the initial_max_streams_bidi QUIC transport parameter as defined in [I-D.ietf-quic-transport].

After the QUIC connection is established, the client can start using the datagram or the stream mode. The client may use PCP [RFC6887] to request the concentrator to accept inbound connections on their behalf. After the negotiation of such port mappings, the concentrator can start opening bidirectional streams to forward inbound connections as well as sending IP packets containing inbound UDP connections in QUIC datagrams.

7. Messages format

In the following sections, we specify the format of each message introduced in this document. They are encoded as TLVs, i.e. (Type, Length, Value) tuples, as illustrated in Figure 4. All TLV fields are encoded in network-byte order.

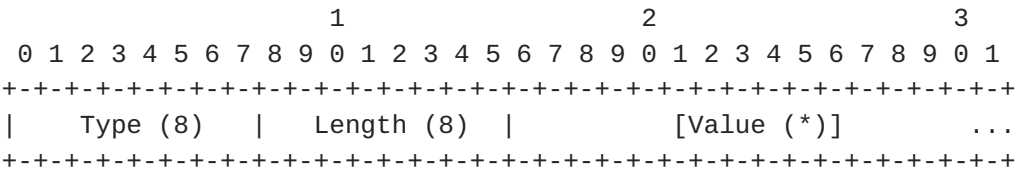


Figure 4: QUIC tunnel TLV Format

The Type field is encoded as a byte and identifies the type of the TLV. The Length field is encoded as a byte and indicate the length of the Value field. A value of zero indicates that no Value field is present. The Value field is a type-specific value whose length is determined by the Length field.

7.1. QUIC tunnel stream TLVs

When using the stream mode, a one or more messages are used to trigger and confirm the establishment of a connection towards the final destination for a given stream. Those messages are exchanged on this given QUIC stream before the TCP connection bytestream. This section describes the format of these messages.

This document specifies the following QUIC tunnel stream TLVs:

Type	Size	Name
0x00	20 bytes	TCP Connect TLV
0x01	38 bytes	TCP Extended Connect TLV
0x02	2 bytes	TCP Connect OK TLV
0x03	Variable	Error TLV
0xff	2 bytes	End TLV

Figure 5: QUIC tunnel stream TLVs

The TCP Connect TLV is used to establish a TCP connection through the tunnel towards the final destination. The TCP Extended Connect TLV allows indicating more information in the establishment request. The TCP Connect OK TLV confirms the establishment of this TCP connection. The Error TLV is used to indicate any out-of-band error that occurred during the TCP connection establishment associated to the QUIC stream. Finally, the End TLV marks the end of the series of TLVs and the start of the bytestream on a given QUIC stream. These TLVs are detailed in the following sections.

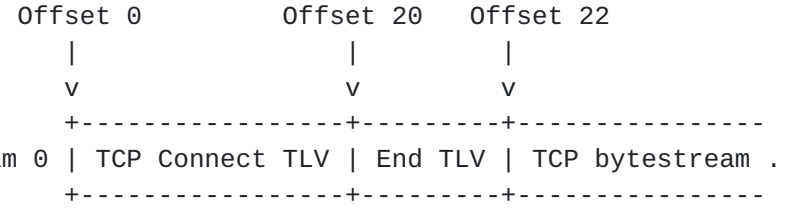


Figure 6: Example of use of QUIC tunnel stream TLVs

7.1.1. TCP Connect TLV

The TCP Connect TLV indicates the final destination of the TCP connection associated to a given QUIC stream. The fields Remote Peer Port and Remote Peer IP Address contain the destination port number and IP address of the final destination.

The Remote Peer IP Address MUST be encoded as an IPv6 address. IPv4 addresses MUST be encoded using the IPv4-Mapped IPv6 Address format defined in [RFC4291]. Further, the Remote Peer IP address field MUST NOT include multicast, broadcast, and host loopback addresses [RFC6890].

A QUIC tunnel peer MUST NOT send more than one TCP Connect TLV per QUIC stream. A QUIC tunnel peer MUST NOT send a TCP Connect TLV if a TCP Extended Connect TLV was previously sent on a given stream. A

QUIC tunnel peer MUST NOT send a TCP Connect TLV on non-self initiated streams.

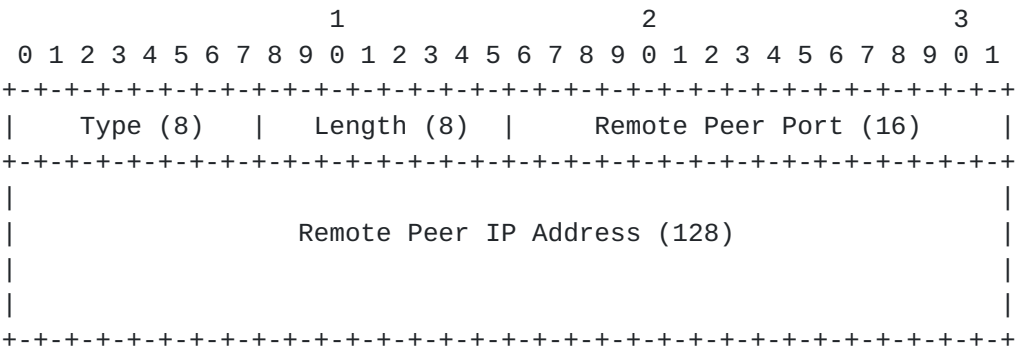


Figure 7: TCP Connect TLV

7.1.2. TCP Extended Connect TLV

The TCP Extended Connect TLV is an extended version of the TCP Connect TLV. It also indicates the source of the TCP connection. The fields Remote Peer Port and Remote Peer IP Address contain the destination port number and IP address of the final destination. The fields Local Peer Port and Local Peer IP Address contain the source port number and IP address of the source of the TCP connection.

The Remote (resp. Local) Peer IP Address MUST be encoded as an IPv6 address. IPv4 addresses MUST be encoded using the IPv4-Mapped IPv6 Address format defined in [RFC4291]. Further, the Remote (resp. Local) Peer IP address field MUST NOT include multicast, broadcast, and host loopback addresses [RFC6890].

A QUIC tunnel peer MUST NOT send more than one TCP Extended Connect TLV per QUIC stream. A QUIC tunnel peer MUST NOT send a TCP Extended Connect TLV if a TCP Connect TLV was previously sent on a given stream. A QUIC tunnel peer MUST NOT send a TCP Extended Connect TLV on non-self initiated streams.

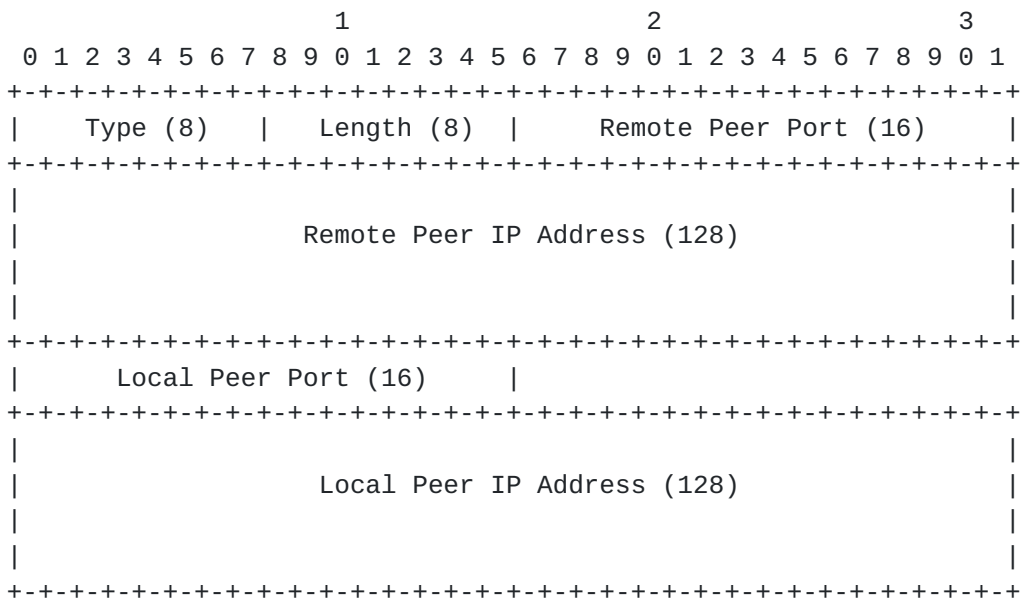


Figure 8: TCP Extended Connect TLV

7.1.3. TCP Connect OK TLV

The TCP Connect OK TLV does not contain a value. Its presence confirms the successful establishment of connection to the final destination. A QUIC peer MUST NOT send a TCP Connect OK TLV on self-initiated streams.

7.1.4. Error TLV

The Error TLV indicates out-of-band errors that occurred during the establishment of the connection to the final destination. These errors can be ICMP Destination Unreachable messages for instance. In this case the ICMP packet received by the concentrator is copied inside the Error Payload field.

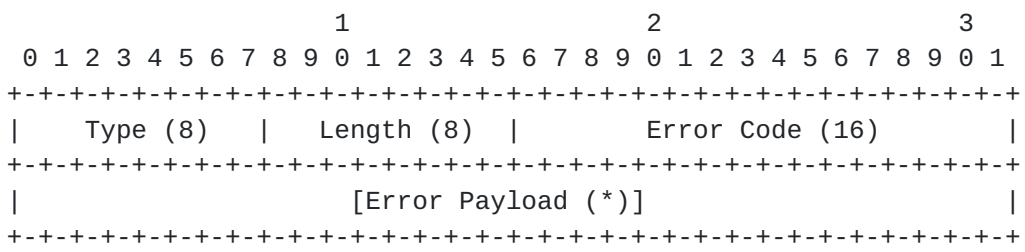


Figure 9: Error TLV

The following bytestream-level error codes are defined in this document:

Code	Name
0x0	Protocol Violation
0x1	ICMP Packet Received
0x2	Malformed TLV
0x3	Network Failure

Figure 10: Bytestream-level Error Codes

- Protocol Violation (0x0): A general error code for all non-conforming behaviors encountered. A QUIC tunnel peer SHOULD use a more specific error code when possible.

- ICMP Packet Received (0x1): This code indicates that the concentrator received an ICMP packet while trying to create the associated TCP connection. The Error Payload contains the packet.

- Malformed TLV (0x2): This code indicates that a received TLV was not successfully parsed or formed. A peer receiving a Connect TLV with an invalid IP address MUST send an Error TLV with this error code.

- Network Failure (0x3): This codes indicates that a network failure prevented the establishment of the connection.

After sending one or more Error TLVs, the sender MUST send an End TLV and terminate the stream, i.e. set the FIN bit after the End TLV.

7.1.5. End TLV

The End TLV does not contain a value. Its existence signals the end of the series of TLVs. The next byte in the QUIC stream after this TLV is the start of the tunneled bytestream.

8. Example flows

This section illustrates the different messages described previously and how they are used in a QUIC tunnel connection. For QUIC STREAM frames, we use the following syntax: STREAM[ID, Stream Data [, FIN]]. The first element is the Stream ID, the second is the Stream Data contained in the frame and the last one is optional and indicates that the FIN bit is set.

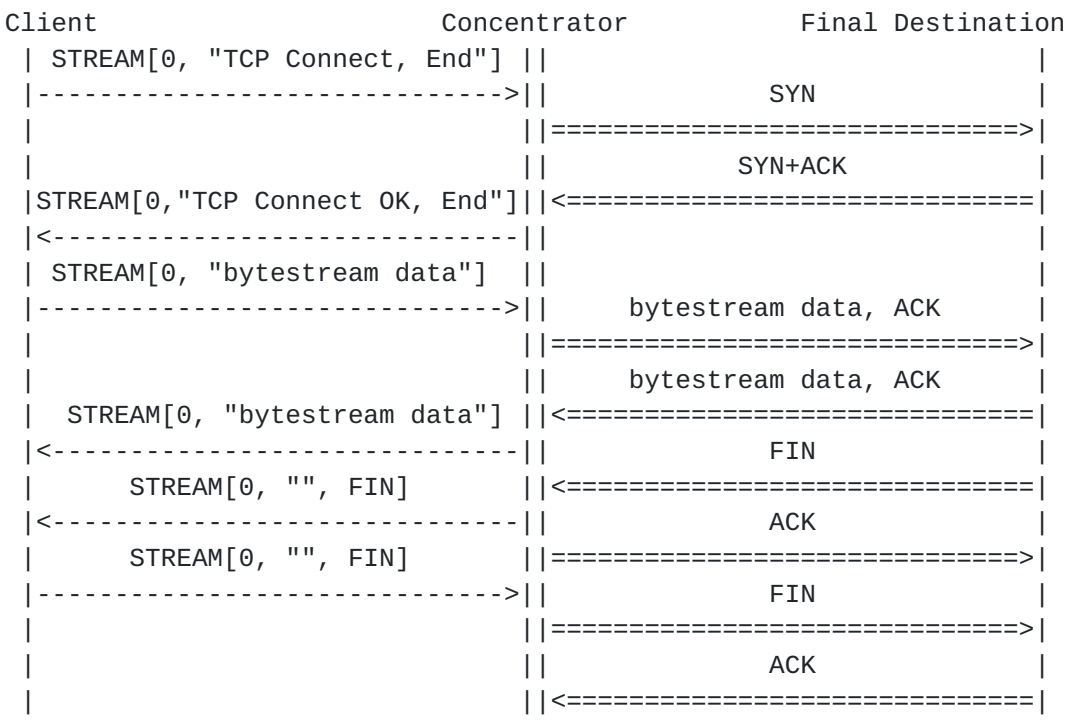


Figure 11: Example flow for the stream mode

On Figure 11, the Client is initiating a TCP connection in stream mode to the Final Destination. A request and a response are exchanged, then the connection is torn down gracefully. A remote-initiated connection accepted by the concentrator on behalf of the client would have the order and the direction of all messages reversed.

9. Security Considerations

9.1. Privacy

The Concentrator has access to all the packets it processes. It MUST be protected as a core IP router, e.g. as specified in [RFC1812].

9.2. Ingress Filtering

Ingress filtering policies MUST be enforced at the network boundaries, i.e. as specified in [RFC2827].

9.3. Denial of Service

There is a risk of an amplification attack when the Concentrator sends a TCP SYN in response of a TCP Connect TLV. When a TCP SYN is larger than the client request, the Concentrator amplifies the client traffic. To mitigate such attacks, the Concentrator SHOULD rate limit the number of pending TCP Connect from a given client.

10. IANA Considerations

10.1. Registration of QUIC tunnel Identification String

This document creates a new registration for the identification of the QUIC tunnel protocol in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [RFC7301].

The "qt" string identifies the QUIC tunnel protocol.

Protocol: QUIC tunnel

Identification Sequence: 0x71 0x74 ("qt")

Specification: This document

10.2. QUIC tunnel stream TLVs

IANA is requested to create a new "QUIC tunnel stream Parameters" registry.

The following subsections detail new registries within "QUIC tunnel stream Parameters" registry.

10.2.1. QUIC tunnel stream TLVs Types

IANA is request to create the "QUIC tunnel stream TLVs Types" sub-registry. New values are assigned via IETF Review (Section 4.8 of [RFC8126]).

The initial values to be assigned at the creation of the registry are as follows:

Code	Name	Reference
0	TCP Connect TLV	[This-Doc]
1	TCP Extended Connect TLV	[This-Doc]
2	TCP Connect OK TLV	[This-Doc]
3	Error TLV	[This-Doc]
255	End TLV	[This-Doc]

10.2.2. QUIC tunnel streams TLVs Error Types

IANA is request to create the "QUIC tunnel stream TLVs Error Types" sub-registry. New values are assigned via IETF Review (Section 4.8 of [RFC8126]).

The initial values to be assigned at the creation of the registry are as follows:

Code	Name	Reference
0	Protocol Violation	[This-Doc]
1	ICMP packet received	[This-Doc]
2	Malformed TLV	[This-Doc]
3	Network Failure	[This-Doc]

11. References

11.1. Normative References

[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>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[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>.

11.2. Informative References

[CoNEXT] De Coninck, Q. and O. Bonaventure, "Multipath QUIC: Design and Evaluation", Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2017) , December 2017.

[I-D.deconinck-quic-multipath] Coninck, Q. and O. Bonaventure, "Multipath Extensions for QUIC (MP-QUIC)", draft-deconinck-quic-multipath-03 (work in progress), August 2019.

[I-D.ietf-lpwan-ipv6-static-context-hc] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and J. Zuniga, "Static Context Header Compression (SCHC) and fragmentation for LPWAN, application to UDP/IPv6", draft-ietf-lpwan-ipv6-static-context-hc-22 (work in progress), October 2019.

[I-D.ietf-quic-transport] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-23 (work in progress), September 2019.

[I-D.ietf-tcpm-converters] Bonaventure, O., Boucadair, M., Gundavelli, S., Seo, S., and B. Hesmans, "0-RTT TCP Convert Protocol", draft-ietf-tcpm-converters-13 (work in progress), October 2019.

[I-D.pauly-quic-datagram] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", draft-pauly-quic-datagram-04 (work in progress), October 2019.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <https://www.rfc-editor.org/info/rfc1812>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/info/rfc2827>.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, DOI 10.17487/RFC3095, July 2001, <https://www.rfc-editor.org/info/rfc3095>.

[RFC3843] Jonsson, L-E. and G. Pelletier, "RObust Header Compression (ROHC): A Compression Profile for IP", RFC 3843, DOI 10.17487/RFC3843, June 2004, <https://www.rfc-editor.org/info/rfc3843>.

[RFC4019] Pelletier, G., "RObust Header Compression (ROHC): Profiles for User Datagram Protocol (UDP) Lite", RFC 4019, DOI 10.17487/RFC4019, April 2005, <https://www.rfc-editor.org/info/rfc4019>.

[RFC4815] Jonsson, L-E., Sandlund, K., Pelletier, G., and P. Kremer, "RObust Header Compression (ROHC): Corrections and Clarifications to RFC 3095", RFC 4815, DOI 10.17487/RFC4815, February 2007, <https://www.rfc-editor.org/info/rfc4815>.

[RFC6846] Pelletier, G., Sandlund, K., Jonsson, L-E., and M. West, "RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)", RFC 6846, DOI 10.17487/RFC6846, January 2013, <https://www.rfc-editor.org/info/rfc6846>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[SIGCOMM19]
De Coninck, Q., Michel, F., Piraux, M., Given-Wilson, T., Legay, A., Pereira, O., and O. Bonaventure, "Pluginizing QUIC", Proceedings of the ACM Special Interest Group on Data Communication , August 2019.

Acknowledgments

Thanks to Quentin De Coninck and Francois Michel for their comments and the proofreading of the first version of this document. Thanks to Gregory Vander Schueren for his comments on the first version of this document.

Authors' Addresses

Maxime Piraux (editor)
UCLouvain

Email: maxime.piraux@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: olivier.bonaventure@uclouvain.be