Workgroup: Internet Area Working Group
Internet-Draft:
draft-piraux-intarea-quic-tunnel-tcp-00
Published: 2 November 2020
Intended Status: Experimental
Expires: 6 May 2021
Authors: M. Piraux   O. Bonaventure
UCLouvain   UCLouvain

Tunneling TCP inside QUIC

Abstract

This document specifies a new operating mode for a QUIC tunnel to efficiently convey TCP bytestreams.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
2. Conventions and Definitions
3. The stream mode
4. Connection establishment
5. Messages format
   5.1. QUIC tunnel stream TLVs
      5.1.1. TCP Connect TLV
      5.1.2. TCP Connect OK TLV
      5.1.3. Error TLV
      5.1.4. End TLV
6. Example flows
7. Security Considerations
   7.1. Denial of Service
8. IANA Considerations
   8.1. QUIC tunnel stream TLVs
      8.1.1. QUIC tunnel stream TLVs Types
      8.1.2. QUIC tunnel streams TLVs Error Types
   8.2. QUIC Transport Parameter Registry
9. References
   9.1. Normative References
   9.2. Informative References
Appendix A. Change Log
   A.1. Since draft-piraux-quic-tunnel-tcp-01
   A.2. Since draft-piraux-quic-tunnel-tcp-00
Acknowledgments
Authors' Addresses

1. Introduction

The recently proposed QUIC tunnel protocol [I-D.piraux-intarea-quic-tunnel] supports the exchange of IP packets and Ethernet frames over a QUIC connection. Its two existing operating modes transports plain packets inside QUIC frames. Their main advantage is that they support any network-layer protocol. 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 to the IP/UDP/QUIC headers of the QUIC connection. For TCP connections for instance, the per-packet overhead can be large.

In this document, we propose a new operating mode for the QUIC tunnel protocol, called the stream mode. It takes advantage of the QUIC streams to efficiently transport TCP bytestreams over a QUIC connection. Section 3 describes this new mode. Section 5 specifies the format of the messages introduced by this document. Section 6 contains example flows.

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. The stream mode

Since QUIC supports 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 QUIC connection. For this, we base our approach on the 0-RTT Convert 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.

A similar approach is used in the 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 5.1. Upon reception of such a TLV message, the concentrator opens a TCP connection towards the specified destination and connects the incoming bytestream of the QUIC connection to the bytestream of the new TCP connection (and similarly in the opposite direction).

Figure 1 summarizes how the new TCP connection is mapped to the QUIC stream. Actions and events of a TCP connection are translated to actions and events of the associated QUIC stream, so that a state transition on 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 1: TCP connection to QUIC stream mapping

When sending STOP_SENDING or RESET_STREAM frames in response to the receipt of a TCP RST, QUIC tunnel peers MUST use the application protocol error code 0x00 (TCP_CONNECTION_RESET).

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

4. Connection establishment

The support of the stream mode is negotiated during the connection establishment by including the quic_tunnel_stream_mode transport parameter (value TBD). The parameter value has no meaning and SHOULD be null.

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

5. Messages format

In the following sections, we specify the format of each message introduced in this document. They are encoded using the TLV format described in [I-D.piraux-intarea-quic-tunnel].

5.1. QUIC tunnel stream TLVs

When using the stream mode, 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 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	2 bytes	TCP Connect OK TLV
0x02	Variable	Error TLV
0xff	2 bytes	End TLV

Figure 2: QUIC tunnel stream TLVs

The TCP Connect TLV is used to request the establishment a TCP connection by the concentrator towards the final destination. The TCP Connect OK TLV confirms the establishment of this TCP connection. The Error TLV is used to indicate any error that occurred during the establishment of a TCP connection. 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.

Future versions of this document may define new TLVs. The End TLV allows a QUIC tunnel peer to send several TLVs before the start of the bytestream.

```
      Offset 0         Offset 20   Offset 22
         |                 |         |
Client   v                 v         v
         +-----------------+---------+----------------
Stream 0 | TCP Connect TLV | End TLV | TCP bytestream ...
         +-----------------+---------+----------------
```

Figure 3: Example of use of QUIC tunnel stream TLVs

Figure 3 illustrates an example of use of QUIC tunnel streams TLVs. In this example, the client opens Stream 0 and sends two TLVs. The first one requests the concentrator to establish a new TCP connection. The second TLV marks the end of the series of TLV and the start of the TCP bytestream.

5.1.1. 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 on non-self initiated streams.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |   Length (8)  |     Remote Peer Port (16)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                  Remote Peer IP Address (128)                 |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: TCP Connect TLV

5.1.2. TCP Connect OK TLV

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

5.1.3. Error TLV

The Error TLV indicates out-of-band errors that occurred during the establishment of the TCP 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.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type (8)   |   Length (8)  |        Error Code (16)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      [Error Payload (*)]                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: 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 6: 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 TCP 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.

5.1.4. 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 part of of the tunneled bytestream.

6. 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.

```
Client                      Concentrator           Final Destination
 | STREAM[0, "TCP Connect, End"] ||                               |
 |------------------------------>||              SYN              |
 |                               ||==============================>|
 |                               ||            SYN+ACK            |
 |STREAM[0,"TCP Connect OK, End"]||<==============================|
 |<------------------------------||                               |
 | STREAM[0, "bytestream data"]  ||                               |
 |------------------------------>||     bytestream data, ACK      |
 |                               ||==============================>|
 |                               ||     bytestream data, ACK      |
 |  STREAM[0, "bytestream data"] ||<==============================|
 |<------------------------------||              FIN              |
 |      STREAM[0, "", FIN]       ||<==============================|
 |<------------------------------||              ACK              |
 |      STREAM[0, "", FIN]       ||==============================>|
 |------------------------------>||              FIN              |
 |                               ||==============================>|
 |                               ||              ACK              |
 |                               ||<==============================|

Legend:
   --- QUIC connection
   === TCP connection
```

Figure 7: Example flow for the stream mode

On Figure 7, 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.

7. Security Considerations

7.1. 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.

8. IANA Considerations

8.1. 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.

8.1.1. 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 Connect OK TLV	[This-Doc]
2	Error TLV	[This-Doc]
255	End TLV	[This-Doc]

8.1.2. 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]

8.2. QUIC Transport Parameter Registry

This document defines a new transport parameter for the negotiation of the stream mode. The following entry in Table 1 should be added to the "QUIC Transport Parameters" registry under the "QUIC Protocol" heading.

Value	Parameter Name	Specification
TBD	quic_tunnel_stream_mode	Section 4

Table 1: Addition to QUIC Transport Parameters Entries

9. References

9.1. 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>.

[I-D.piraux-intarea-quic-tunnel] Piraux, M., Bonaventure, O., and A. Masputra, "Tunneling Internet protocols inside QUIC", Work in Progress, Internet-Draft, draft-piraux-intarea-quic-tunnel-00, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-piraux-intarea-quic-tunnel-00.txt>.

[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>.

9.2. Informative References

[I-D.ietf-tcpm-converters] Bonaventure, O., Boucadair, M., Gundavelli, S., Seo, S., and B. Hesmans, "0-RTT TCP Convert Protocol", Work in Progress, Internet-Draft, draft-ietf-tcpm-converters-19, 22 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tcpm-converters-19.txt>.

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

[I-D.piraux-quic-tunnel]
Piraux, M., Bonaventure, O., and A. Masputra, "Tunneling Internet protocols inside QUIC", Work in Progress, Internet-Draft, draft-piraux-quic-tunnel-03, 12 August 2020, <http://www.ietf.org/internet-drafts/draft-piraux-quic-tunnel-03.txt>.

[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>.

Appendix A. Change Log

A.1. Since draft-piraux-quic-tunnel-tcp-01

* Nits

A.2. Since draft-piraux-quic-tunnel-tcp-00

* Add the quic_tunnel_stream_mode transport parameter for negotiation

Acknowledgments

This documents draws heavily on the initial version of [I-D.piraux-quic-tunnel]. Their contributors are thanked again here. This work was partially supported by the MQUIC project.

Authors' Addresses

Maxime Piraux
UCLouvain

Email: maxime.piraux@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: olivier.bonaventure@uclouvain.be