QUIC
Internet-Draft
Intended status: Standards Track
Expires: 11 December 2020

M. Thomson
Mozilla

9 June 2020

Version-Independent Properties of QUIC
draft-ietf-quic-invariants-09

Abstract

This document defines the properties of the QUIC transport protocol that are expected to remain unchanged over time as new versions of the protocol are developed.

Note to Readers

Discussion of this draft takes place on the QUIC working group mailing list (quic@ietf.org (mailto:quic@ietf.org)), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=quic.

Working Group information can be found at https://github.com/quicwg; source code and issues list for this draft can be found at https://github.com/quicwg/base-drafts/labels/-invariants.

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 December 2020.

Copyright Notice

Table of Contents

1. Introduction

In addition to providing secure, multiplexed transport, QUIC [QUIC-TRANSPORT] includes the ability to negotiate a version. This allows the protocol to change over time in response to new requirements. Many characteristics of the protocol will change between versions.

This document describes the subset of QUIC that is intended to remain stable as new versions are developed and deployed. All of these invariants are IP-version-independent.

The primary goal of this document is to ensure that it is possible to deploy new versions of QUIC. By documenting the properties that can't change, this document aims to preserve the ability to change any other aspect of the protocol. Thus, unless specifically described in this document, any aspect of the protocol can change between different versions.

Appendix A is a non-exhaustive list of some incorrect assumptions that might be made based on knowledge of QUIC version 1; these do not apply to every version of QUIC.

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.

This document uses terms and notational conventions from [QUIC-TRANSPORT].

3. An Extremely Abstract Description of QUIC

QUIC is a connection-oriented protocol between two endpoints. Those endpoints exchange UDP datagrams. These UDP datagrams contain QUIC packets. QUIC endpoints use QUIC packets to establish a QUIC connection, which is shared protocol state between those endpoints.

4. Notational Conventions

Packet diagrams in this document use a format defined in [QUIC-TRANSPORT] to illustrate the order and size of fields.

Complex fields are named and then followed by a list of fields surrounded by a pair of matching braces. Each field in this list is separated by commas.

Individual fields include length information, plus indications about fixed value, optionality, or repetitions. Individual fields use the following notational conventions, with all lengths in bits:

x (A): Indicates that x is A bits long

x (A..B): Indicates that x can be any length from A to B; A can be omitted to indicate a minimum of zero bits and B can be omitted to indicate no set upper limit; values in this format always end on an octet boundary

x (?) = C: Indicates that x has a fixed value of C

x (E) ...: Indicates that x is repeated zero or more times (and that each instance is length E)

This document uses network byte order (that is, big endian) values. Fields are placed starting from the high-order bits of each byte.

Figure 1 shows an example structure:

```
Example Structure {
  One-bit Field (1),
  7-bit Field with Fixed Value (7) = 61,
  Arbitrary-Length Field (..),
  Variable-Length Field (8..24),
  Repeated Field (8) ...,
}
```

Figure 1: Example Format

5. QUIC Packet Headers

QUIC endpoints exchange UDP datagrams that contain one or more QUIC packets. This section describes the invariant characteristics of a QUIC packet. A version of QUIC could permit multiple QUIC packets in a single UDP datagram, but the invariant properties only describe the first packet in a datagram.

QUIC defines two types of packet header: long and short. Packets with long headers are identified by the most significant bit of the first byte being set; packets with a short header have that bit cleared.

Aside from the values described here, the payload of QUIC packets is version-specific and of arbitrary length.

5.1. Long Header

Long headers take the form described in Figure 2.

```
Long Header Packet {
  Header Form (1) = 1,
  Version-Specific Bits (7),
  Version (32),
  Destination Connection ID Length (8),
  Destination Connection ID (0..2040),
  Source Connection ID Length (8),
  Source Connection ID (0..2040),
  Version-Specific Data (..),
}
```

Figure 2: QUIC Long Header

A QUIC packet with a long header has the high bit of the first byte set to 1. All other bits in that byte are version specific.

The next four bytes include a 32-bit Version field. Versions are described in Section 5.4.

The next byte contains the length in bytes of the Destination Connection ID field that follows it. This length is encoded as an 8-bit unsigned integer. The Destination Connection ID field follows the Destination Connection ID Length field and is between 0 and 255 bytes in length. Connection IDs are described in Section 5.3.

The next byte contains the length in bytes of the Source Connection ID field that follows it. This length is encoded as a 8-bit unsigned integer. The Source Connection ID field follows the Source Connection ID Length field and is between 0 and 255 bytes in length.

The remainder of the packet contains version-specific content.

5.2. Short Header

Short headers take the form described in Figure 3.

```
Short Header Packet {
  Header Form (1) = 0,
  Version-Specific Bits (7),
  Destination Connection ID (..),
  Version-Specific Data (..),
}
```

Figure 3: QUIC Short Header

A QUIC packet with a short header has the high bit of the first byte set to 0.

A QUIC packet with a short header includes a Destination Connection ID immediately following the first byte. The short header does not include the Connection ID Lengths, Source Connection ID, or Version fields. The length of the Destination Connection ID is not encoded in packets with a short header and is not constrained by this specification.

The remainder of the packet has version-specific semantics.

5.3. Connection ID

A connection ID is an opaque field of arbitrary length.

The primary function of a connection ID is to ensure that changes in addressing at lower protocol layers (UDP, IP, and below) don't cause packets for a QUIC connection to be delivered to the wrong QUIC endpoint. The connection ID is used by endpoints and the intermediaries that support them to ensure that each QUIC packet can be delivered to the correct instance of an endpoint. At the endpoint, the connection ID is used to identify which QUIC connection the packet is intended for.

The connection ID is chosen by each endpoint using version-specific methods. Packets for the same QUIC connection might use different connection ID values.

5.4. Version

QUIC versions are identified with a 32-bit integer, encoded in network byte order. Version 0 is reserved for version negotiation (see Section 6). All other version numbers are potentially valid.

The properties described in this document apply to all versions of QUIC. A protocol that does not conform to the properties described in this document is not QUIC. Future documents might describe additional properties which apply to a specific QUIC version, or to a range of QUIC versions.

6. Version Negotiation

A QUIC endpoint that receives a packet with a long header and a version it either does not understand or does not support might send a Version Negotiation packet in response. Packets with a short header do not trigger version negotiation.

A Version Negotiation packet sets the high bit of the first byte, and thus it conforms with the format of a packet with a long header as defined in Section 5.1. A Version Negotiation packet is identifiable as such by the Version field, which is set to 0x00000000.

```
Version Negotiation Packet {
  Header Form (1) = 1,
  Unused (7),
  Version (32) = 0,
  Destination Connection ID Length (8),
  Destination Connection ID (0..2040),
  Source Connection ID Length (8),
  Source Connection ID (0..2040),
  Supported Version (32) ...,
}
```

Figure 4: Version Negotiation Packet

The Version Negotiation packet contains a list of Supported Version fields, each identifying a version that the endpoint sending the packet supports. The Supported Version fields follow the Version field. A Version Negotiation packet contains no other fields. An endpoint MUST ignore a packet that contains no Supported Version fields, or a truncated Supported Version.

Version Negotiation packets do not use integrity or confidentiality protection. A specific QUIC version might authenticate the packet as part of its connection establishment process.

An endpoint MUST include the value from the Source Connection ID field of the packet it receives in the Destination Connection ID field. The value for Source Connection ID MUST be copied from the Destination Connection ID of the received packet, which is initially randomly selected by a client. Echoing both connection IDs gives clients some assurance that the server received the packet and that the Version Negotiation packet was not generated by an off-path attacker.

An endpoint that receives a Version Negotiation packet might change the version that it decides to use for subsequent packets. The conditions under which an endpoint changes QUIC version will depend on the version of QUIC that it chooses.

See [QUIC-TRANSPORT] for a more thorough description of how an endpoint that supports QUIC version 1 generates and consumes a Version Negotiation packet.

7. Security and Privacy Considerations

It is possible that middleboxes could use traits of a specific version of QUIC and assume that when other versions of QUIC exhibit similar traits the same underlying semantic is being expressed. There are potentially many such traits (see Appendix A). Some effort has been made to either eliminate or obscure some observable traits in QUIC version 1, but many of these remain. Other QUIC versions might make different design decisions and so exhibit different traits.

The QUIC version number does not appear in all QUIC packets, which means that reliably extracting information from a flow based on version-specific traits requires that middleboxes retain state for every connection ID they see.

The Version Negotiation packet described in this document is not integrity-protected; it only has modest protection against insertion by off-path attackers. QUIC versions MUST define a mechanism that authenticates the values it contains.

8. IANA Considerations

This document makes no request of IANA.

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>.

[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

[QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using Transport Layer Security (TLS) to Secure QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-tls-29, 9 June 2020, <https://tools.ietf.org/html/draft-ietf-quic-tls-29>.

[QUIC-TRANSPORT] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-29, 9 June 2020, <https://tools.ietf.org/html/draft-ietf-quic-transport-29>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

Appendix A. Incorrect Assumptions

There are several traits of QUIC version 1 [QUIC-TRANSPORT] that are not protected from observation, but are nonetheless considered to be changeable when a new version is deployed.

This section lists a sampling of incorrect assumptions that might be made based on knowledge of QUIC version 1. Some of these statements are not even true for QUIC version 1. This is not an exhaustive list, it is intended to be illustrative only.

The following statements are NOT guaranteed to be true for every QUIC version:

* QUIC uses TLS [QUIC-TLS] and some TLS messages are visible on the wire

* QUIC long headers are only exchanged during connection establishment

* Every flow on a given 5-tuple will include a connection establishment phase

* The first packets exchanged on a flow use the long header

* The last packet before a long period of quiescence might be assumed to contain only an acknowledgment

* QUIC uses an AEAD (AEAD_AES_128_GCM [RFC5116]) to protect the packets it exchanges during connection establishment

* QUIC packet numbers are encrypted and appear as the first encrypted bytes

* QUIC packet numbers increase by one for every packet sent

* QUIC has a minimum size for the first handshake packet sent by a client

* QUIC stipulates that a client speaks first

* QUIC packets always have the second bit of the first byte (0x40) set

* A QUIC Version Negotiation packet is only sent by a server

* A QUIC connection ID changes infrequently

* QUIC endpoints change the version they speak if they are sent a Version Negotiation packet

* The version field in a QUIC long header is the same in both directions

* Only one connection at a time is established between any pair of QUIC endpoints

Author’s Address

Martin Thomson
Mozilla

Email: mt@lowentropy.net