Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 28, 2018

M. Kuehlewind
B. Trammell
ETH Zurich

October 25, 2017

Manageability of the QUIC Transport Protocol
draft-ietf-quic-manageability-01

Abstract

This document discusses manageability of the QUIC transport protocol, focusing on caveats impacting network operations involving QUIC traffic. Its intended audience is network operators, as well as content providers that rely on the use of QUIC-aware middleboxes, e.g. for load balancing.

Status of This Memo

This Internet-Draft will expire on April 28, 2018.

Copyright Notice

Table of Contents

1. Introduction

QUIC [QUIC] is a new transport protocol currently under development in the IETF quic working group, focusing on support of semantics as needed for HTTP/2 [QUIC-HTTP]. Based on current deployment practices, QUIC is encapsulated in UDP and encrypted by default. The current version of QUIC integrates TLS [QUIC-TLS] to encrypt all payload data and most control information. Given QUIC is an end-to-end transport protocol, all information in the protocol header, even that which can be inspected, is is not meant to be mutable by the

network, and will therefore be integrity-protected to the extent possible.

This document provides guidance for network operation on the management of QUIC traffic. This includes guidance on how to interpret and utilize information that is exposed by QUIC to the network as well as explaining requirement and assumptions that the QUIC protocol design takes toward the expected network treatment. It also discusses how common network management practices will be impacted by QUIC.

Of course, network management is not a one-size-fits-all endeavour: practices considered necessary or even mandatory within enterprise networks with certain compliance requirements, for example, would be impermissible on other networks without those requirements. This document therefore does not make any specific recommendations as to which practices should or should not be applied; for each practice, it describes what is and is not possible with the QUIC transport protocol as defined.

QUIC is at the moment very much a moving target. This document refers the state of the QUIC working group drafts as well as to changes under discussion, via issues and pull requests in GitHub current as of the time of writing.

1.1. Notational Conventions

The words "MUST", "MUST NOT", "SHOULD", and "MAY" are used in this document. It's not shouting; when these words are capitalized, they have a special meaning as defined in [RFC2119].

2. Features of the QUIC Wire Image

In this section, we discusses those aspects of the QUIC transport protocol that have an impact on the design and operation of devices that forward QUIC packets. Here, we are concerned primarily with QUIC's unencrypted wire image, which we define as the information available in the packet header in each QUIC packet, and the dynamics of that information. Since QUIC is a versioned protocol, also the wire image of the header format can change. However, at least the mechanism by which a receiver can determine which version is used and the meaning and location of fields used in the version negotiation process need to be fixed.

This document is focused on the protocol as presently defined in [QUIC] and [QUIC-TLS], and will change to track those documents.

2.1. QUIC Packet Header Structure

The QUIC packet header is under active development; see section 5 of [QUIC] for the present header structure.

The first bit of the QUIC header indicates the present of a long header that exposes more information than the short header. The long header is used during connection start including version negotiation, server retry, and 0-RTT data while the short header is used after the handshake and therefore on most data packets to limited unnecessary header overhead. The fields and location of these fields as defined by the current version of QUIC for the long header are fixed for all future version as well. However, note that future versions of QUIC may provide additional fields. In the current version of quic the long header for all header types has a fixed length, containing, besides the Header Form bit, a 7-bit header Type, a 64-bit Connection ID, a 32-bit Packet Number, and a 32-bit Version. The short header is variable length where bits after the Header Form bit indicate the present on the Connection ID, and the length of the packet number.

The following information may be exposed in the packet header:

- header type: the long header has a 7-bit header type field following the Header Form bit. The current version of QUIC defines 6 header types, namely Version Negotiation, Client Initial, Server Stateless Retry, Server Cleartext, Client Cleartext, 0-RTT Protected.

- connection ID: The connection ID is always present on the long and optionally present on the short header indicated by the Connection ID Flag. If present at the short header it at the same position then for the long header. The position and length pf the congestion ID itself as well as the Connection ID flag in the short header is fixed for all versions of QUIC. The connection ID identifies the connection associated with a QUIC packet, for load-balancing and NAT rebinding purposes; see Section 4.3 and Section 2.3. Therefore it is also expected that the Connection ID will either be present on all packets of a flow or none of the short header packets. However, this field is under endpoint control and there is no protocol mechanism that hinders the sending endpoint to revise its decision about exposing the Connection ID at any time during the connection.

- packet number: Every packet has an associated packet number. The packet number increases with each packet, and the least-significant bits of the packet number are present on each packet. In the short header the length of the exposed packet number field

is defined by the (short) header type and can either be 8, 16, or 32 bits. See Section 2.4.

- version number: The version number is present on the long headers and identifies the version used for that packet, expect for the Version negotiation packet. The version negotiation packet is fixed for all version of QUIC and contains a list of versions that is supported by the sender. The version in the version field of the Version Negotiation packet is the reflected version of the Client Initial packet and is therefore explicitly n ot supported by the sender.

- key phase: The short header further has a Key Phase flag that is used by the endpoint identify the right key that was used to encrypt the packet.

2.2. Integrity Protection of the Wire Image

As soon as the cryptographic context is established, all information in the QUIC header, including those exposed in the packet header, is integrity protected. Further, information that were sent and exposed in previous packets when the cryptographic context was established yet, e.g. for the cryptographic initial handshake itself, will be validated later during the cryptographic handshake, such as the version number. Therefore, devices on path MUST NOT change any information or bits in QUIC packet headers. As alteration of header information would cause packet drop due to a failed integrity check at the receiver, or can even lead to connection termination.

2.3. Connection ID and Rebinding

The connection ID in the QUIC packer header is used to allow routing of QUIC packets at load balancers on other than five-tuple information, ensuring that related flows are appropriately balanced together; and to allow rebinding of a connection after one of the endpoint's addresses changes - usually the client's, in the case of the HTTP binding. The client set a Connection ID in the Initial Client packet that will be used during the handshake. A new connection ID is then provided by the server during connection establishment, that will be used in the short header after the handshake. Further a server might provide additional connection IDs that can the used by the client at any time during the connection. Therefore if a flow changes one of its IP addresses it may keep the same connection ID, or the connection ID may also change together with the IP address migration, avoiding linkability; see Section 7.6 of [QUIC].

2.4. Packet Numbers

The packet number field is always present in the QUIC packet header. The packet number exposes the least significant 32, 16, or 8 bits of an internal packet counter per flow direction that increments with each packet sent. This packet counter is initialized with a random 31-bit initial value at the start of a connection.

Unlike TCP sequence numbers, this packet number increases with every packet, including those containing only acknowledgment or other control information. Indeed, whether a packet contains user data or only control information is intentionally left unexposed to the network. The packet number increases with every packet but they sender may skip packet numbers.

While loss detection in QUIC is based on packet numbers, congestion control by default provides richer information than vanilla TCP does. Especially, QUIC does not rely on duplicated ACKs, making it more tolerant of packet re-ordering.

2.5. Initial Handshake and PMTUD

[Editor's note: text needed.]

2.6. Version Negotiation and Greasing

Version negotiation is not protected, given the used protection mechanism can change with the version. However, the choices provided in the list of version in the Version Negotiation packet will be validated as soon as the cryptographic context has been established. Therefore any manipulation of this list will be detected and will cause the endpoints to terminate the connection.

Also note that the list of versions in the Version Negotiation packet may contain reserved versions. This mechanism is used to avoid ossification in the implementation on the selection mechanism. Further, a client may send a Initial Client packet with a reserved version number to trigger version negotiation. In the Version Negotiation packet the connection ID and packet number of the Client Initial packet are reflected to provide a proof of return-routability. Therefore changing these information will also cause the connection to fail.

3. Network-visible information about QUIC flows

This section addresses the different kinds of observations and inferences that can be made about QUIC flows by a passive observer in the network based on the wire image in Section 2. Here we assume a

bidirectional observer (one that can see packets in both directions in the sequence in which they are carried on the wire) unless noted.

3.1. Identifying QUIC traffic

The QUIC wire image is not specifically designed to be distinguishable from other UDP traffic.

The only application binding currently defined for QUIC is HTTP [QUIC-HTTP]. HTTP over QUIC uses UDP port 443 by default, although URLs referring to resources available over HTTP over QUIC may specify alternate port numbers. Simple assumptions about whether a given flow is using QUIC based upon a UDP port number may therefore not hold; see also [RFC7605] section 5.

3.1.1. Identifying Negotiated Version

An in-network observer assuming that a set of packets belongs to a QUIC flow can infer the version number in use by observing the handshake: a Client Initial with a given version followed by Server Cleartext packet with the same version implies acceptance of that version.

Negotiated version cannot be identified for flows for which a handshake is not observed, such as in the case of NAT rebinding; however, these flows can be associated with flows for which a version has been identified; see Section 3.3.

In the rest of this section, we discuss only packets belonging to Version 1 QUIC flows, and assume that these packets have been identified as such through the observation of a version negotiation.

3.1.2. Rejection of Garbage Traffic

A related question is whether a first packet of a given flow on known QUIC-associated port is a valid QUIC packet, in order to support in-network filtering of garbage UDP packets (reflection attacks, random backscatter). While heuristics based on the first byte of the packet (packet type) could be used to separate valid from invalid first packet types, the deployment of such heuristics is not recommended, as packet types may have different meanings in future versions of the protocol.

3.2. Connection confirmation

Connection establishment requires cleartext packets and is using a TLS handshake on stream 0. Therefore it is detectable using heuristics similar to those used to detect TLS over TCP. 0-RTT

connection may additional also send data packets, right after the Client Initial with the TLS client hello. These data may be reordered in the network, therefore it may be possible that 0-RTT Protected data packets are seen before the Client Initial packet.

3.3. Flow association

The QUIC Connection ID (see Section 2.3) is designed to allow an on-path device such as a load-balancer to associate two flows as identified by five-tuple when the address and port of one of the endpoints changes; e.g. due to NAT rebinding or server IP address migration. An observer keeping flow state can associate a connection ID with a given flow, and can associate a known flow with a new flow when when observing a packet sharing a connection ID and one endpoint address (IP address and port) with the known flow.

The connection ID to be used for a long-running flow is chosen by the server (see [QUIC] section 5.6) during the handshake. This value should be treated as opaque; see Section 4.3 for caveats regarding connection ID selection at servers.

3.4. Flow teardown

The QUIC does not expose the end of a connection; the only indication to on-path devices that a flow has ended is that packets are no longer observed. Stateful devices on path such as NATs and firewalls must therefore use idle timeouts to determine when to drop state for QUIC flows.

Changes to this behavior are currently under discussion: see https://github.com/quicwg/base-drafts/issues/602.

3.5. Round-trip time measurement

Round-trip time of QUIC flows can be inferred by observation once per flow, during the handshake, as in passive TCP measurement. The delay between the Client Initial packet and the Server Cleartext packet sent back to the client represents the RTT component on the path between the observer and the server, and the delay between this packet and the Client Cleartext packet in reply represents the RTT component on the path between the observer and the client. This measurement necessarily includes any application delay at both sides. Note that the Server's reply mayalso be a Version Negotiation or Server Stateless Retry packet. In this case the Client will send another Client Initial or the connection will fail.

The lack of any acknowledgement information or timestamping information in the QUIC wire image makes running passive RTT estimation impossible.

Changes to this behavior are currently under discussion: see https://github.com/quicwg/base-drafts/issues/631.

3.6. Packet loss measurement

All QUIC packets carry packet numbers in cleartext, and while the protocol allows packet numbers to be skipped, skipping is not recommended in the general case. This allows the trivial one-sided estimation of packet loss and reordering between the sender and a given observation point ("upstream loss"). However, since retransmissions are not identifiable as such, loss between an observation point and the receiver ("downstream loss") cannot be reliably estimated.

3.7. Flow symmetry measurement

QUIC explicitly exposes which side of a connection is a client and which side is a server during the handshake. In addition, the symmerty of a flow (whether primarily client-to-server, primarily server-to-client, or roughly bidirectional, as input to basic traffic classification techniques) can be inferred through the measurement of data rate in each direction. While QUIC traffic is protected and ACKS may be padded, padding is not required.

4. Specific Network Management Tasks

In this section, we address specific network management and measurement techniques and how QUIC's design impacts them.

4.1. Stateful treatment of QUIC traffic

Stateful treatment of QUIC traffic is possible through QUIC traffic and version identification (Section 3.1) and observation of the handshake for connection confirmation (Section 3.2). The lack of any visible end-of-flow signal (Section 3.4) means that this state must be purged either through timers or through least-recently-used eviction, depending on application requirements.

4.2. Passive network performance measurement and troubleshooting

Extremely limited loss and RTT measurement are possible by passive observation of QUIC traffic; see Section 3.5 and Section 3.6.

4.3. Server cooperation with load balancers

In the case of content distribution networking architectures including load balancers, the connection ID provides a way for the server to signal information about the desired treatment of a flow to the load balancers.

Server-generated Connection IDs must not encode any information other that that needed to route packets to the appropriate backend server(s): typically the identity of the backend server or pool of servers, if the data-center's load balancing system keeps "local" state of all flows itself. Care must be exercised to ensure that the information encoded in the Connection ID is not sufficient to identify unique end users. Note that by encoding routing information in the Connection ID, load balancers open up a new attack vector that allows bad actors to direct traffic at a specific backend server or pool. It is therefore recommended that Server-Generated Connection ID includes a cryptographic MAC that the load balancer pool server are able to identify and discard packets featuring an invalid MAC.

4.4. DDoS Detection and Mitigation

Current practices in detection and mitigation of Distributed Denial of Service (DDoS) attacks generally involve passive measurement using network flow data [RFC7011], classification of traffic into "good" (productive) and "bad" (DoS) flows, and filtering of these bad flows in a "scrubbing" environment. Key to successful DDoS mitigation is efficient classification of this traffic.

Limited first-packet garbage detection as in Section 3.1.2 and stateful tracking of QUIC traffic as in Section 4.1 above can be used in this classification step. For traffic where the classification step did not observe a QUIC handshake, the presence of packets carrying the same Connection ID in both directions is a further indication of legitimate traffic. Note that these classification techniques help only against floods of garbage traffic, not against DDoS attacks using legitimate QUIC clients.

Note that the use of a connection ID to support connection migration renders 5-tuple based filtering insufficient, and requires more state to be maintained by DDoS defense systems. However, it is questionable if connection migrations needs to be supported in a DDOS attack. If the connection migration is not visible to the network that performs the DDoS detection, an active, migrated QUIC connection may be blocked by such a system under attack. However, a defense system might simply rely on the fast resumption mechanism provided by QUIC. See also https://github.com/quicwg/base-drafts/issues/203

4.5. QoS support and ECMP

[EDITOR'S NOTE: this is a bit speculative; keep?]

QUIC does not provide any additional information on requirements on Quality of Service (QoS) provided from the network. QUIC assumes that all packets with the same 5-tuple {dest addr, source addr, protocol, dest port, source port} will receive similar network treatment. That means all stream that are multiplexed over the same QUIC connection require the same network treatment and are handled by the same congestion controller. If differential network treatment is desired, multiple QUIC connections to the same server might be used, given that establishing a new connection using 0-RTT support is cheap and fast.

QoS mechanisms in the network MAY also use the connection ID for service differentiation, as a change of connection ID is bound to a change of address which anyway is likely to lead to a re-route on a different path with different network characteristics.

Given that QUIC is more tolerant of packet re-ordering than TCP (see Section 2.4), Equal-cost multi-path routing (ECMP) does not necessarily need to be flow based. However, 5-tuple (plus eventually connection ID if present) matching is still beneficial for QoS given all packets are handled by the same congestion controller.

5. IANA Considerations

This document has no actions for IANA.

6. Security Considerations

Supporting manageability of QUIC traffic inherently involves tradeoffs with the confidentiality of QUIC's control information; this entire document is therefore security-relevant.

Some of the properties of the QUIC header used in network management are irrelevant to application-layer protocol operation and/or user privacy. For example, packet number exposure (and echo, as proposed in this document), as well as connection establishment exposure for 1-RTT establishment, make no additional information about user traffic available to devices on path.

At the other extreme, supporting current traffic classification methods that operate through the deep packet inspection (DPI) of application-layer headers are directly antithetical to QUIC's goal to provide confidentiality to its application-layer protocol(s); in these cases, alternatives must be found.

7. Contributors

Dan Druta contributed text to Section 4.4. Igor Lubashev contributed text to Section 4.3 on the use of the connection ID for load balancing.

8. Acknowledgments

This work is partially supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI), and by the Swiss State Secretariat for Education, Research, and Innovation under contract no. 15.0268. This support does not imply endorsement.

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>.

9.2. Informative References

[Ding2015] Ding, H. and M. Rabinovich, "TCP Stretch Acknowledgments and Timestamps - Findings and Impliciations for Passive RTT Measurement (ACM Computer Communication Review)", July 2015, <http://www.sigcomm.org/sites/default/files/ccr/papers/2015/July/0000000-0000002.pdf>.

[IPIM] Allman, M., Beverly, R., and B. Trammell, "In-Protocol Internet Measurement (arXiv preprint 1612.02902)", December 2016, <https://arxiv.org/abs/1612.02902>.

[QUIC] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-07 (work in progress), October 2017.

[QUIC-HTTP] Bishop, M., "Hypertext Transfer Protocol (HTTP) over QUIC", draft-ietf-quic-http-07 (work in progress), October 2017.

[QUIC-TLS]
Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-07 (work in progress), October 2017.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC7605] Touch, J., "Recommendations on Using Assigned Transport Port Numbers", BCP 165, RFC 7605, DOI 10.17487/RFC7605, August 2015, <https://www.rfc-editor.org/info/rfc7605>.

Authors' Addresses

Mirja Kuehlewind
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: mirja.kuehlewind@tik.ee.ethz.ch

Brian Trammell
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch