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

M. Kuehlewind
B. Trammell
ETH Zurich

October 25, 2017

Applicability of the QUIC Transport Protocol
draft-ietf-quic-applicability-01

Abstract

This document discusses the applicability of the QUIC transport protocol, focusing on caveats impacting application protocol development and deployment over QUIC. Its intended audience is designers of application protocol mappings to QUIC, and implementors of these application protocols.

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 http://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 April 28, 2018.

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] such as stream-multiplexing to avoid head-of-line blocking. Based on current deployment practices, QUIC is encapsulated in UDP. The version of QUIC that is currently under development will integrate TLS 1.3 [TLS13] to encrypt all payload data and most control information.

This document provides guidance for application developers that want to use the QUIC protocol without implementing it on their own. This includes general guidance for application use of HTTP/2 over QUIC as well as the use of other application layer protocols over QUIC. For specific guidance on how to integrate HTTP/2 with QUIC, see [QUIC-HTTP].

In the following sections we discuss specific caveats to QUIC's applicability, and issues that application developers must consider when using QUIC as a transport for their application.

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. The Necessity of Fallback

QUIC uses UDP as a substrate for userspace implementation and port numbers for NAT and middlebox traversal. While there is no evidence of widespread, systematic disadvantage of UDP traffic compared to TCP in the Internet [Edeline16], somewhere between three [Trammell16] and five [Swett16] percent of networks simply block UDP traffic. All applications running on top of QUIC must therefore either be prepared to accept connectivity failure on such networks, or be engineered to fall back to some other transport protocol. This fallback SHOULD provide TLS 1.3 or equivalent cryptographic protection, if available, in order to keep fallback from being exploited as a downgrade attack. In the case of HTTP, this fallback is TLS 1.3 over TCP.

These applications must operate, perhaps with impaired functionality, in the absence of features provided by QUIC not present in the fallback protocol. For fallback to TLS over TCP, the most obvious difference is that TCP does not provide stream multiplexing and therefore stream multiplexing would need to be implemented in the application layer if needed. Further, TCP without the TCP Fast Open extension does not support 0-RTT session resumption. TCP Fast Open can be requested by the connection initiator but might no be supported by the far end or could be blocked on the network path. Note that there is some evidence of middleboxes blocking SYN data even if TFO was successfully negotiated (see [PaaschNanog]).

Any fallback mechanism is likely to impose a degradation of performance; however, fallback MUST not silently violate the application's expectation of confidentiality or integrity of its payload data.

Moreover, while encryption (in this case TLS) is inseparable integrated with QUIC, TLS negotiation over TCP can be blocked. In case it is RECOMMENDED to abort the connection, allowing the application to present a suitable prompt to the user that secure communication is unavailable.

3. Zero RTT

QUIC provides for 0-RTT connection establishment (see section 3.2 of [QUIC]). This presents opportunities and challenges for applications using QUIC.

3.1. Thinking in Zero RTT

A transport protocol that provides 0-RTT connection establishment to recently contacted servers is qualitatively different than one that does not from the point of view of the application using it. Relative tradeoffs between the cost of closing and reopening a connection and trying to keep it open are different; see Section 3.3.

Applications must be slightly rethought in order to make best use of 0-RTT resumption. Most importantly, application operations must be divided into idempotent and non-idempotent operations, as only idempotent operations may appear in 0-RTT packets. This implies that the interface between the application and transport layer exposes idempotence either ecplicitly or implicitly.

3.2. Here There Be Dragons

Retransmission or (malicious) replay of data contained in 0-RTT resumption packets could cause the server side to receive two copies of the same data. This is further described in [HTTP-RETRY]. Data sent during 0-RTT resumption also cannot benefit from perfect forward secrecy (PFS).

Data in the first flight sent by the client in a connection established with 0-RTT MUST be idempotent (as specified in section 3.2 in [QUIC-TLS]). Applications MUST be designed, and their data MUST be framed, such that multiple reception of idempotent data is recognized as such by the receiverApplications that cannot treat data that may appear in a 0-RTT connection establishment as idempotent MUST NOT use 0-RTT establishment. For this reason the QUIC transport SHOULD provide an interface for the application to indicate if 0-RTT support is in general desired or a way to indicate whether data is idempotent, and/or whether PFS is a hard requirement for the application.

3.3. Session resumption versus Keep-alive

[EDITOR'S NOTE: see https://github.com/quicwg/ops-drafts/issues/6]

4. Use of Streams

QUIC's stream multiplexing feature allows applications to run multiple streams over a single connection, without head-of-line blocking between streams, associated at a point in time with a single five-tuple. Stream data is carried within Frames, where one (UDP) packet on the wired can carry one of multiple stream frames.

Stream can be independently open and closed, gracefully or by error. If a critical stream for the application is closed, the application can generate respective error messages on the application layer to inform the other end or the higher layer and eventually indicate quic to reset the connection. QUIC, however, does not need to know which streams are critical, and does not provide an interface to exceptional handling of any stream. There are special streams in QUIC that are used for control on the QUIC connection, however, these streams are not exposed to the apllication.

Mapping of application data to streams is application-specific and described for HTTP/s in [QUIC-HTTP]. In general data that can be processed independently, and therefore would suffer from head of line blocking, if forced to be received in order, should be transmitted over different streams. If there is a logical grouping of those data chunks or messages, stream can be reused, or a new stream can be opened for each chunk/message. However, a QUIC receiver has a maximum number of concurrently open streams. If the stream limit is exhausted a sender is able to indicate that more streams are needed, however, this does not automatically lead to an increase of the maximum number of streams by the receiver. Therefore it can be valuable to expose this maximum number to the application, or the number of currently still available, unused streams, and make the mapping of data to streams dependent on this information.

Further, streams have a maximum number of bytes that can be sent on one stream. This number is high enough (2^64) that this will usually not be reached with current applications. Applications that send chunks of data over a very long period of time (such as days, months, or years), should rather utilize the 0-RTT seesion resumption ability provided by QUIC, than trying to maintain one connection open.

4.1. Stream versus Flow Multiplexing

Streams are meaningful only to the application; since stream information is carried inside QUIC's encryption boundary, no information about the stream(s) whose frames are carried by a given packet is visible to the network. Therefore stream multiplexing is not intended to be used for differentiating streams in terms of network treatment. Application traffic requiring different network treatment SHOULD therefore be carried over different five-tuples (i.e. multiple QUIC connections). Given QUIC's ability to send application data in the first RTT of a connection (if a previous connection to the same host has been successfully established to provide the respective credentials), the cost for establishing another connection are extremely low.

4.2. Paketization and latency

Quic provides an interface that provides multiple streams to the application, however, the application usually doesn't have control how the data transmitted over one stream is mapped into frame and how frames are bundled into packets. By default QUIC will try to maximally pack packets to minimize bandwidth consumption and computational costs with one or multiple same data frames. If not enough data available to send QUIC may even wait for a short time, trading of latency and bandwidth effeciency. This time might either be pre-configured or can the dynamically adjusted based on the observed sending pattern of the application. If the apllication requires low latency, with only small chunks of data to send, it may be valuable to indicate to QUIC that all data should be send out immediately. Or if a certain sending pattern is know by the application, it might also provide valuabe to QUIC how long it should wait to bundle frame into a packet.

4.3. Prioritization

Stream prioritization is not exposed to the network, nor to the receiver. Prioritization can be realized by the sender and the QUIC transport should provide an interface for applications to prioritize streams [QUIC]. Further applications can implement their own prioritization scheme on top of QUIC: an an (application) protocol that run on top of QUIC can define explict messages for signaling priority, such as those defined for HTTP/2; it can define rules that allow an endpoint to determine priority based on context; or it can provide a higher level interface and leave the determination to the application on top.

Priority handling of retransmissions can be implemented by the sender in the transport layer. [QUIC] recommends to retransmit lost data before new data, unless indicated differently by the application. Currently QUIC only provides fully reliable stream transmission, and as such prioritization of retransmissionis likely beneficial in most cases, as gaps that get filled up and thereby free up flow control. For not fully reliable streams priority scheduling of retransmissions over data of higher-priority streams might not be desired. In this case QUIC could also provide an interface or derive the prioritization decision from the reliability level of the stream.

5. Graceful connection closure

[EDITOR'S NOTE: give some guidance here about the steps an application should take; however this is still work in progress]

6. Information exposure and the Connection ID

QUIC exposes some information to the network in the unencrypted part of the header, either before the encryption context is established, because the information is intended to be used by the network. QUIC has a long header that is used during connection establishment and for other control processes, and a short header that may be used for data transmission in an established connection. While the long header is fixed and exposes some information, the short header only exposes the packet number by default and may optionally expose a connection ID.

Given that exposing this information may make it possible to associate multiple addresses with a single client during rebinding, which has privacy implications, an application may indicate to not support exposure of certain information after the handshake. Specificially, an application that has additional information that the client is not behind a NAT and the server is not behind a load balancer, and therefore it is unlikely that the addresses will be re-bound, may indicate to the transport that is wishes to not expose a connection ID.

6.1. Server-Generated Connection ID

QUIC supports a server-generated Connection ID, transmitted to the client during connection establishment: see Section 5.7 of [QUIC] Servers behind load balancers should propose a Connection ID during the handshake, encoding the identity of the server or information about its load balancing pool, in order to support stateless load balancing. Once the server generates a Connection ID that encodes its identity, every CDN load balancer would be able to forward the packets to that server without needing information about every specific flow it is forwarding.

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.

6.2. Using Server Retry for Redirection

QUIC provide a Server Retry packet that can be send by a server in response to the Client Initial packet. The server may choose a new connection ID in that packet and the client will retry by sending another Client Initial packet with the server-selected connection ID. This mechanism can be used to redirect a connection to a different server, e.g. due to performance reasons or when servers in a server pool are upgraded gradually, and therefore may support different versions of QUIC. In this case, it is assumed that all servers belonging to a certain pool are served in cooperation with load balancers that forward the traffic based on the connection ID. A server can chose the connection ID in the Server Retry packet such that the load balancer will redirect the next Client Initial packet to a different server in that pool.

7. Use of Versions and Cryptographic Handshake

Versioning in QUIC may change the the protocol's behavior completely, except for the meaning of a few header fields that have been declared to be fixed. As such version of QUIC with a higher version number does not necessarily provide a better service, but might simply provide a very different service, so an application needs to be able to select which versions of QUIC it wants to use.

A new version could use an encryption scheme other than TLS 1.3 or higher. [QUIC] specifies requirements for the cryptographic handshake as currently realized by TLS 1.3 and described in a separate specification [QUIC-TLS]. This split is performed to enable light-weight versioning with different cryptographic handshakes.

8. IANA Considerations

This document has no actions for IANA.

9. Security Considerations

See the security considerations in [QUIC] and [QUIC-TLS]; the security considerations for the underlying transport protocol are relevant for applications using QUIC, as well.

Application developers should note that any fallback they use when QUIC cannot be used due to network blocking of UDP SHOULD guarantee the same security properties as QUIC; if this is not possible, the connection SHOULD fail to allow the application to explicitly handle fallback to a less-secure alternative. See Section 2.

10. Contributors

Igor Lubashev contributed text to Section 6 on server-selected connection IDs.

11. 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.

12. References

12.1. Normative References

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

[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>.

[TLS13] Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-07 (work in progress), October 2017.

12.2. Informative References

[Edeline16] Edeline, K., Kuehlewind, M., Trammell, B., Aben, E., and B. Donnet, "Using UDP for Internet Transport Evolution (arXiv preprint 1612.07816)", December 2016, <https://arxiv.org/abs/1612.07816>.

[HTTP-RETRY] Nottingham, M., "Retrying HTTP Requests", draft-nottingham-httpbis-retry-01 (work in progress), February 2017.

[I-D.nottingham-httpbis-retry]
Nottingham, M., "Retrying HTTP Requests", draft-nottingham-httpbis-retry-01 (work in progress), February 2017.

[PaaschNanog]
Paasch, C., "Network Support for TCP Fast Open (NANOG 67 presentation)", June 2016, <https://www.nanog.org/sites/default/files/Paasch_Network_Support.pdf>.

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

[Swett16] Swett, I., "QUIC Deployment Experience at Google (IETF96 QUIC BoF presentation)", July 2016, <https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf>.

[Trammell16]
Trammell, B. and M. Kuehlewind, "Internet Path Transparency Measurements using RIPE Atlas (RIPE72 MAT presentation)", May 2016, <https://ripe72.ripe.net/wp-content/uploads/presentations/86-atlas-udpdiff.pdf>.

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