Workgroup: QUIC
Internet-Draft: draft-huitema-quic-in-space-00
Published: 24 September 2023
Intended Status: Informational
Expires: 27 March 2024
Authors: C. Huitema
Private Octopus Inc.
M. Blanchet
Viagenie

QUIC in Space

Abstract

This document discusses the challenges of running the QUIC transport over deep space links, where delays are in order of minutes and communications are based on scheduled time windows. Using the experience of various testbeds, it provides guidance to implementations to support this use case. This document may apply to other use cases that have similar characteristics, such as IoT in disconnected and distant settings.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://huitema.github.io/quic-in-space/draft-huitema-quic-in-space.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-huitema-quic-in-space/.

Discussion of this document takes place on the QUIC Working Group mailing list (mailto:quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/. Subscribe at https://www.ietf.org/mailman/listinfo/quic/.

Source for this draft and an issue tracker can be found at https://github.com/huitema/quic-in-space.

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 27 March 2024.

Table of Contents

1. Introduction

QUIC is a new transport bringing very interesting features that could enable its use in space, where TCP is not good, as assessed in [DTN-ARCH]. However, QUIC was designed for terrestrial Internet, which brings assumptions on typical delays and connectivity. In

(deep) space, delays are much larger, in order of minutes (4-20 minutes to Mars), and long disruptions, such as because of orbital dynamics, in order of minutes or hours or days.

It may be possible to modify the base behavior of QUIC stacks to satisfy these requirements. For example, several assumptions, such as initial RTT, are just static constants in the code that could be externalized so they could better start the QUIC machinery in the context of space.

The purpose of this document is to provide guidance for supporting space communications in QUIC implementations. It should be noted that it may also apply to other use cases that have similar characteristics, such as IoT in disconnected and far away settings, but these are not considered specifically in this document.

Various other considerations such as how to store packets during disruptions or network management considerations are out of scope of this document.

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. Timer Constants in QUIC

QUIC implementations typically use a number of time related variables, with different characteristics. Some may be constants suggested by the QUIC specification or chosen by the stack developers, some may be negotiated between peers, and some may be discovered as part of running the protocol. Preliminary tests showed that setting the constant values appropriately seemed to make QUIC usable in those deep space scenarios. Therefore, this document discusses them in the following sections.

3.1. Probe Timeout and Initial RTT (#initial-rtt)

As defined in [QUIC-RECOVERY], QUIC uses two mechanisms to detect packet losses. The acknowledgement based method detect losses if a packet is not yet acknowledged while packets sent later have already been. That method works well even if the transmission delay is long, but cannot detect the loss of the "last packet". For that, QUIC uses the probe timeout defined in Section 6.2 of [QUIC-RECOVERY]. If the last packet is not yet acknowledged after the probe timeout, the endpoint sends a "probe" to trigger an acknwledgement. The probe timer is initially set as a function of the measured RTT and RTTVAR.

It is then increased exponentially as the number of unacknowledged repetitions increases.

The mechanism works well if the transmission delay is long after the RTT has been evaluated at least once. But before that, the probe timeout is set as a function of the Initial RTT, whose recommended value is 333 milliseconds per Section 6.2.2 of [QUIC-RECOVERY]. Many implementations use smaller values because waiting too long results in longer connection delays when losses occur. The recommended initial value of 333ms results in a PTO of 1 second, but the shorter values used by some implementations can result in a PTO of 200 or 250ms. On a long delay link, we will probably see the following:

1. Initial transmission

2. First repeat after timer (e.g. 1 second)

3. 2nd repeat after timer (e.g., 1 second again), after which the timer is doubled

4. 3rd repeat after the increased (e.g., 2 seconds), after which the timer is doubled again

5. etc.

If we let the process go long enough, a succession of doubling will probably match the required value, probably after a dozen repeats if the delay is about 20 minutes. In that case, one of the dozen repeats will most likely be successful, but of course a lot of extra energy will have been expanded. But the connection establishment will fail if the process is interrupted too soon, either because the maximum number of repeats has been reached, or because the "idle timeout" has been exceeded.

3.2. Server Side Timeout (#server-timeout)

The long delays also affect the server side of the handshake. The server will only be able to assess the RTT of the initial path when it receives the first acknowledgement from the client. The handshake will fail if the server discards the connection state before receiving this first acknowledgement. As on the client side, this could happen if the maximum number of repeats has been reached, or if the "idle timeout" has been exceeded.

3.3. Idle Timeout

The idle timeout is defined in Section 10.1 of [QUIC-TRANSPORT]. Each peer proposes a "max_idle_timeout" value, and commits to close the connection if no activity happens during that timeout. The "max_idle_timeout" value is often set as a constant by either the

stack or the application using it, with typical values ranging from a few seconds to a few minutes.

The specification anticipated the usage of long delay links somewhat, and states that "To avoid excessively small idle timeout periods, endpoints **MUST** increase the idle timeout period to be at least three times the current Probe Timeout (PTO)." This will prevent interference between the idle timeout once the PTO has been properly assessed. However, this proper assessment of the PTO requires properly assessing the RTT, which is requires a full RTT.

If the Initial Idle Timeout, Initial RTT and Initial PTO are set values too small, the connection attempt will be interrupted by the Idle Timeout and will fail.

Since the idle timeout is negotiated as the minimum of the values proposed by the two endpoints, both peers should proposed values that allow for a successful handshake.

4. Flow Control

Flow control in QUIC allow an endpoint to limit how many many bytes the peer can send on the opened streams, how many bytes the peer can send on specific streams, and how many streams the peer can open. Different stacks follow different strategies, balancing two risks:

* if the flow control is too loose, the peer could send data faster than the local application can process and create a form of local congestion.

* if the flow control is too restrictive, the peer will be blocked and will have to wait for the next flow control update before sending data or opening streams.

The peer will not be blocked if three conditions are met:

* the "max streams" credits allow for a number of stream at least as large as expected to be open in an RTT.

* the global flow control (MAX DATA) allows transmission of a full bandwidth-delay product (BDP) worth of data,

* for each stream, the stream flow control allows transmission of either a full BDP worth of data, or the full size of the data stream.

Setting these three limits correctly requires anticipating the RTT and bandwidth of the connection. Implementations relying on adaptive algorithms are at risk here, especially if they use a low default value until RTT and bandwidth have been measured.

5. Congestion control and Slow Start

QUIC implementation use congestion control to ensure that they are not sending data faster than the network path can forward, which would cause network queues and packet drops. Congestion control will typically set a congestion window size (CWIN) to limit the amount of bytes in transit. Transmission will be slowed down CWIN is lower than the BDP.

The main congestion control issues for long delay space connections are observed at the beginning of the connection. The congestion control algorithms typically start with conservative values of CWIN, which they ramp up progressively, typically not faster than one doubling per RTT. On a long delay space connection, this ramping up will take a very long time, leading to very inefficient use of the connection.

Implementations have tried to palliate this issue in many ways:

* measure the transmission speed of short trains of packets to rapidly estimate the bandwidth of the connection,

* remember the delays and bandwidths of past connections and set the initial parameters of new connections accordingly,

* obtain estimates of delays and bandwidth through network management interfaces and use them to set appropriate parameters.

If initial parameters are set correctly, connections can still be unnecessarily throttled if they fail to adapt to changing conditions. For example, after the initial "slow start", the classic RENO algorithm moves to a "congestion avoidance" phase in which the CWIN increases by at most one packet per RTT -- which would be completely inadequate with RTTs of several minutes.

5.1. Setting the initial BDP

The congestion control issues are not specific to the very long delays of space communications. They are visible on paths through geostationary satellites. The recommended approach is to somehow remember path characteristics from previous connections, then use the remembered values to speed up the start-up phase of congestion control.

The "careful resume" draft suggest a cautious approach of only using the remembered BDP values after the RTT has been verified, see [I-D.tsvwg-careful-resume]. This verification takes one RTT, which is a tradeoff between the desire to ramp up transmission rate promptly and the risk of causing congestion on the transmission path if the remembered value exceeds the current path characteristics.

If the BDP is remembered and set, the value can also be used to set flow control parameters as mentioned in Section 4.

6. Packet losses

Packet losses will likely occur in space communications like in other media. The loss recovery mechanisms spedified in [QUIC-RECOVERY] correct losses after at best one RTT, with repeated losses requiring further RTT sized delays. In space communications, the long delays for packet loss recovery will affect the responsiveness of the application. In some cases, the loss recovery delays may extend the duration of the transfer past the time of availability of the transmission links.

6.1. Packet Losses During Handshake

Packet losses during the initial handshake may prevent the endpoints from obtaining a proper estimate of the RTT. The implementations in these situations can either repeat the packets "too soon" if they underestimate the RTT, or "too late" if they set a large value. Expereince show that of those too pitfalls, "too soon" is much better, because it simply leads to repeating too many copies of the handshake packets. This may look wasteful, but it does ensure that the handshake completes rapidly even if some packets are actually lost.

6.2. Reordering Buffers

If packets carrying stream data are lost, the other packets received on that stream are supposed to be buffered until the packet loss is corrected. Receivers have to be ready to buffer a full BDP worth of data. This means not only having large amount of receive buffers, but also make sure that reordering of data is done efficiently, risking otherwise some significant performance bottlenecks.

Reordering will also interfere with per-stream and per-connection flow control. In Section 4, we wrote that the amount of credits should be at least one full BDP. But if we assume that a full BDP worth of buffers can be consumed by reordering after losses, then the required credits are actually two BDPs, not just one.

6.3. Using Forward Error Correction

The effect of packet losses could be alleviated by using some form of Forward Error Correction. This is an active research issue, see for example [I-D.michel-quic-fec]

7. Further studies

There are other considerations related to using QUIC in space, related to naming, security, or the management of multiple paths.

Most QUIC stack can set connections towards specified IP addresses, but most existing QUIC applications rely on the DNS to find the IP addresses associated to names of servers. Using DNS in space probably poses its own set of challenges, which deserve their own study.

Per [QUIC-TLS], QUIC embeds TLS 1.3 and relies on it to negotiate security keys. This document discusses the effects of long delays on the initial handshake, which embeds the TLS 1.3 handshake. There are secondary effects that ought to be discussed, such as the handling of certificate verification and possibly certificate revocations, or the extra roundtrip required for performing client authorization.

It is probably possible to use multiple path to increase the throughput or reliability of transmissions. Operating multiple path with long delays ought to be discussed in a future version of this document.

8. Security Considerations

The solutions envisaged here to alleviate the effect of long delays on QUIC connections may well create some form of security issues. These ought to be discussed in a future version of this document.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[I-D.tsvwg-careful-resume] "*** BROKEN REFERENCE ***".

[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/rfc/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/rfc/rfc8174>.

10.2. Informative References

[DTN-ARCH]

Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/rfc/rfc4838>.

[I-D.michel-quic-fec] Michel, F. and O. Bonaventure, "Forward Erasure Correction for QUIC loss recovery", Work in Progress, Internet-Draft, draft-michel-quic-fec-00, 21 October 2022, <https://datatracker.ietf.org/doc/html/draft-michel-quic-fec-00>.

[QUIC-RECOVERY] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[QUIC-TRANSPORT] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Christian Huitema
Private Octopus Inc.

Email: huitema@huitema.net

Marc Blanchet
Viagenie

Email: marc.blanchet@viagenie.ca