Workgroup: Network Working Group
Internet-Draft:
draft-huitema-quic-mpath-option-01
Published: 12 September 2021
Intended Status: Standards Track
Expires: 16 March 2022
Authors: C. Huitema
Private Octopus Inc.

QUIC Multipath Negotiation Option

Abstract

The initial version of QUIC provides support for path migration. We propose a simple mechanism to support not just migration, but also simultaneous usage of multiple paths. In its simplest form, this mechanisms simply requires that multipath senders keep track of which packet is sent on what path, and use that information to manage congestion control and loss recovery. With that, clients can send data on any validated path, and server on any validated path on which the client recently sent non-probing packets. A more sophisticated mechanism can be negotiated to explicitly manage paths and packet scheduling.

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 16 March 2022.

Copyright Notice

Table of Contents

1. Introduction

The QUIC Working Group is debating how much priority should be given to the support of multipath in QUIC. This is not a new debate. The QUIC transport [QUIC-TRANSPORT] includes a number of mechanisms to handle multiple paths, including the support for NAT rebinding and for explicit migration.

The processing of received packets by QUIC nodes only requires that the connection context can be retrieved using the combination of IP addresses and UDP ports in the IP and UDP headers, and connection identifier in the packet header. Once the connection context is identified, the receiving node will verify if the packet can be successfully decrypted. If it is, the packet will be processed and eventually an acknowledgement will inform the sender that it was received.

In theory, that property of QUIC implies that senders could send packets using whatever IP addresses or UDP ports they see fit, as long as they carry a valid connection ID, and are properly encrypted. Senders use the Path Challenge and Response mechanism to verify that the path is valid before sending packets, but even that is optional. The NAT rebinding mechanisms, for example, rely on the possibility of receiving packets without a preliminary Path Challenge. However, in practice, the sender's choice of sending paths is limited by the path migration logic in [QUIC-TRANSPORT].

Path migration according to [QUIC-TRANSPORT] is always initiated by the node in the client role. That node will test the validity of paths using the path challenge response mechanism, and at some point decide to switch all its traffic to a new path. The node in the server role will detect that data traffic (i.e., non probing frames) is sent on a new path, and on detecting that, will switch its own traffic to that new path. After that, client and server can release the resource tied to the old path, for example by retiring the connection identifiers, releasing the memory context used for managing per path congestion, or forgetting the IP addresses and ports associated with that path. By sending data on a new path, the client provides an implicit signal to discard the old path. If the client was to spread data on several paths, the server would probably become confused.

This draft proposes a simple mechanism to enable transmission on several paths simultaneously. In the simplest form, this only requires updating the handling of paths specified in [QUIC-TRANSPORT] by the proposed simple multipath management, see Section 2. As an option, endpoints can also negotiate use of the explicit path management specified in [QUIC-MP-LIU], see Section 2.5.

1.1. Relation with other drafts

This draft shares a common author with [QUIC-MP-LIU], and refers to some of its content. The difference is that this draft maintains the single packet number space for application packets defined in [QUIC-TRANSPORT], instead of using separate number spaces for each path as in [QUIC-MP-LIU]. Both of these drafts have been implemented, and the lessons derived from this implementation exercise are listed in Section 3.

1.2. 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.

2. Simple Multipath Specification

The path management specified in section 9 of [QUIC-TRANSPORT] fulfills multiple goals: it direct a peers to switch sending through a new preferred path, and it allows the peer to release resources associated with the old path. Multipath requires several changes to that mechanism:

- Allow simultaneous transmission of non probing frames on multiple paths.

- Continue using an existing path even if non probing frames have been received on another frame.

- Manage the removal of paths that have been abandoned.

The multipath management is a departure from the specification of path management in section 9 of [QUIC-TRANSPORT]. As such, it requires negotiation between the two endpoints, as specified in Section 2.1. This negotiation will also determine whether the endpoints will use simple or explicit multipath management.

2.1. Enable Simple Multipath Extension

The path management extension is negotiated by means of the "enable_simple_multipath" transport parameter:

- "enable_simple_multipath" (TBD)

The "enable_path_management" transport parameter is included if the endpoint wants to receive or accepts frames for this connection. This parameter is encoded as a variable integer as specified in section 16 of [QUIC-TRANSPORT]. It can take one of the following values:

1. I am able to receive and process data on multiple paths

2. I am able to receive and process explicit path management frames

3. I am able to receive and process explicit path management frames and I would like to send them.

Peers receiving another value SHOULD terminate the connection with a TRANSPORT PARAMETER error.

The simple multipath option is successfully negotiated if both peers propose a non zero value. If the negotiation fail, peers are expected to handle paths as specified in section 9 of [QUIC-TRANSPORT].

A peer that advertises its capability of sending explicit path management frames using option values 3 MUST NOT send these frames if the other peer does not advertise its ability to process them by sending the enable_path_management TP with option 2 or 3. If usages of these frames is negotiated, the peers shall use the explicit path management option defined in Section 2.5.

2.2. Path Creation and Path Validation

When the simple multipath option is negotiated, clients may initiate transmission on multiple paths, using the mechanisms specified in section 8.2 of [QUIC-TRANSPORT] to validate the new paths. After receiving packets from the client on the new paths, the servers may in turn attempt validate these paths using the same mechanisms.

Client may decide to send non-probing packets on validated paths. In contrast with the specification in section 9 of [QUIC-TRANSPORT], the server MUST NOT assume that receiving non-probing packets on a new path indicates an attempt to migrate to that path. Instead, servers SHOULD mark paths over which non-probing packets are received as available for transmission.

2.3. Path Removal

At any time in the connection, each endpoint manages a set of paths that are available for transmission, as explained in Section 2.2. At any time, the client can decide to abandon one of these paths, following for example changes in local connectivity or changes in local preferences. After a client abandons a path, the server will not receive any more non-probing packets on that path.

When only one path is available, servers MUST follow the specifications in [QUIC-TRANSPORT]. When more than one path is available, servers shall monitor the arrival of non-probing packets on the available paths. Servers SHOULD stop sending traffic on paths through which non-probing packet was received in the last 3 path RTTs, but MAY ignore that rule if it would disqualify all available paths. Server MAY release the resource associated with paths for which no non-probing packet was received for a sufficiently long path-idle delay, but SHOULD only release resource for the last available path if no traffic is received for the duration of the idle timeout, as specified in section 10.1 of [QUIC-TRANSPORT]. Server implementations manage the value of the path-idle delay as a trade-off between keeping resource such as Connection Identifiers in use for an excessive time, and having to promptly reestablish a path after a spurious estimate of path abandonment by the client.

2.4. Packet Transmission

Once the connection is complete, if the simple multipath option is negotiated, each endpoint can send 1RTT protected packets on any of the validated paths. The packet sequence numbers are allocated from the common number space, so that for example path number number N could be sent on one path and packet number N+1 on another.

ACK frames report the numbers of packets that have been received so far, regardless of the path on which they have been received.

Senders MUST manage per-path congestion status, and MUST NOT send more data on a given path than congestion control on that path allows. This is already a requirement of [QUIC-TRANSPORT].

In order to implement per path congestion control, the senders maintain an association between previously sent packet numbers and the path over which these packets were sent. When a packet is newly acknowledged, the delay between the transmission of that packet and its first acknowledgement is used to update the RTT statitics for the sending path, and to update the state of the congestion control for that path.

Packet loss detection MUST be adapted to allow for different RTT on different paths. For example, timer computations should take into account the RTT of the path on which a packet was sent. Detections based on packet numbers shall compare a given packet number to the highest packet number received for that path.

2.5. Explicit multipath management

The transport parameter negotiation specified in Section 2.1 allows parties to negotiate the use of explicit multipath management. When validated, this option enables the use of the PATH_STATUS and frames QOE_CONTROL_SIGNALS defined in [QUIC-MP-LIU] to manage the selection of paths and the scheduling of packets on multiple paths. In particular:

* The path identifiers used in PATH_STATUS and QOE_CONTROL_SIGNALS frames SHALL be computed as specified in identify paths as defined in section 4.1 of [QUIC-MP-LIU].

* The PATH_STATUS frames SHALL be used to manage paths as specified in section 4.4 and 4.5 of [QUIC-MP-LIU].

* The packet scheduling MAY use the QOE_CONTROL_SIGNALS frames as defined in section 7 of [QUIC-MP-LIU].

2.6. Negotiation of Simple Multipath or MP QUIC

Nodes may be programmed to support either the simple multipath management defined in this document or the "Multipath Extension for QUIC" defined in [QUIC-MP-LIU]. These options are not compatible, and the negotiation MUST result in the selection of at most one of these options. A QUIC client MAY send transport parameters proposing support for both options, but a QUIC server MUST NOT propose both options. A QUIC client receiving server transport parameters proposing to use both options simultaneously MUST terminate the connection with a TRANSPORT PARAMETER error.

3. Implementation Considerations

The guidance provided in this section is based on early tests and early deployments. We expect that as implementations and deployment progresses, we will gain experience in the development and deployment of the simple QUIC Multipath option.

3.1. Scheduling Transmissions

The Simple Multipath Specification in Section 2 makes a number of recommendations such as only sending on paths that are validated, or heeding per-path congestion control algorithms. It stops short of mandating any specific scheduling algorithm. However, based on early experience, we can make a set of implementations recommendations.

3.1.1. Tying Packet Scheduling and Loss Recovery

In principle, implementations that spread traffic on multiple paths benefit from more bandwidth than using a single path, and are expected to obtain better performance. However, early tests show that losses on one path may cause delays until the lost data is retransmitted and finally received. This can result in increased latency compared to transmission on a single "non lossy" path.

In early tests, these effects were mitigated by tying packet scheduling and loss recovery. If high losses were noticed on one path, the scheduling algorithm would privilege transmission on alternate paths.

Of course, packet losses may be due to transient events. If losses are experienced on a path, it is important to keep probing that path, and to restart scheduling packets on that path if probing is successful. This may be achieved by using redundant transmissions, see Section 3.1.3.

3.1.2. Multipath Scheduling and Slow-Start

Congestion control algorithms often rely on an initial phase such as "slow-start" to assess the capacity of path, see for example section 7.3 of [QUIC-RECOVERY]. The evaluation phase often ends with the detection of packet losses. As discussed in Section 3.1.1, these losses may have an adverse effect on the overall quality of experience.

In early tests, there were some benefits in using redundant transmissions to avoid the adverse effects of packet losses during initial slow-start, see Section 3.1.3.

3.1.3. Redundant Transmissions

Redundant transmission is the process of sending the same data multiple times. In early tests, this process was used to feed redundant packets to paths that recently experienced losses, or to paths undergoing slow start. Instead of probing the "dubious" path with dummy packets made of PING and PAD frames, the implementation could send copies of frames previously transmitted on other paths.

This mechanism is generally safe, because duplication of frames naturally occurs as a side effect of "spurious" retransmissions. Implementations of QUIC routinely filter out duplicate STREAM_DATA content, duplicate stream and flow control management frames, or duplicate acknowledgments. The datagram frames specified in [QUIC-DATAGRAM] are a notable exception; including such frames in redundant packets could well result in duplicate delivery.

Redundant transmission is a trade-off between risking delays due to losses and risking delays due to inefficient use of the transmission capacity. It is not always appropriate. One might also argue that duplicate transmission is a simplistic form of forward error correction, and that a more elaborate form of forward error correction might result in better performance.

3.2. Acknowledgement and Ranges

If senders decide to send packets on paths with different transmission delays, some packets will very probably be received out of order. This will cause the ACK frames to carry multiple ranges of received packets. The large number of range increases the size of ACK frames, causing transmission and processing overhead.

Early tests showed that the size and overhead of the ACK frames could be controlled by the combination of one or several of the following:

* Limiting the number of transmissions of a specific ACK range, on the assumption that a sufficient number of transmissions almost certainly ensures reception by the peer.

* Not transmitting again ACK ranges that were present in an ACK frame acknowledged by the peer.

* Delay acknowledgement to allow for arrival of "hole filling" packets.

* Limit the total number of ranges sent in an ACK frame.

* Send multiple messages for a given path in a single socket operation, so that a series of packets sent from a single path uses a series of consecutive sequence numbers without creating holes.

3.3. Computing Path RTT

Acknowledgement delays are the sum of two one-way delays, the delay on the packet sending path and the delay on the return path chosen for the acknowledgements. When different paths have different characteristics, this can cause acknowledgement delays to vary widely. Consider for example multipath transmission using both a terrestrial path, with a latency of 50ms in each direction, and a geostationary satellite path, with a latency of 300ms in both directions. The acknowledgement delay will depend on the combination of paths used for the packet transmission and the ACK transmission, as shown in Table 1.

ACK Path \ Data path	Terrestrial	Satellite
Terrestrial	100ms	350ms
Satellite	350ms	600ms

Table 1: Example of ACK delays using multiple paths

Using the default algorithm specified in [QUIC-RECOVERY] would result in suboptimal performance, computing average RTT and standard deviation from a series of different delay measurements of different combined paths. At the same time, early tests show that it is desirable to send ACKs through the shortest path, because a shorter ACK delay results in a tighter control loop and better performances. The tests also showed that it is desirable to send copies of the

ACKs on multiple paths, for robustness if a path experience sudden losses.

An early implementation mitigated the delay variation issue by using time stamps, as specified in [QUIC-Timestamp]. When the timestamps are present, the implementation can estimate the transmission delay on each one-way path, and can then use these one way delays for more efficient implementations of recovery and congestion control algorithms.

If timestamps are not available, implementations could estimate one way delays using statistical techniques. For example, in the example shown in Table 1, implementations can use use "same path" measurements to estimate the one way delay of the terrestrial path to about 50ms in each direction, and that of the satellite path to about 300ms. Further measurements can then be used to maintain estimates of one way delay variations, using logical similar to Kalman filters. But statistical processing is error rone, and using time stamps provides more robust measurements.

4. Security Considerations

TBD. There are probably ways to abuse this.

5. IANA Considerations

This document registers a new value in the QUIC Transport Parameter Registry:

Value	Parameter Name.	Specification
TBD (experiments use 0xbab5)	enable_simple_multipath	Section 2.1

Table 2: Addition to QUIC Transport Parameters Entries

6. References

6.1. Normative References

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

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

[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>.

6.2. Informative References

[QUIC-DATAGRAM] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", <https://datatracker.ietf.org/doc/draft-ietf-quic-datagram/>.

[QUIC-MP-LIU] Liu, Y., Ma, Y., Huitema, C., An, Q., and Z. Li, "Multipath Extension for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-recovery, <https://datatracker.ietf.org/doc/draft-liu-multipath-quic/>.

[QUIC-Timestamp] Huitema, C., "QUIC Timestamps For Measuring One-Way Delays", August 2020, <https://datatracker.ietf.org/doc/draft-huitema-quic-ts/>.

Appendix A. Acknowledgements

Author's Address

Christian Huitema
Private Octopus Inc.

Email: huitema@huitema.net