Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2021

C. Huitema
Private Octopus Inc.

October 20, 2020

QUIC Multipath Negotiation Option
draft-huitema-quic-mpath-option-00

Abstract

The initial version of QUIC provides support for path migration. In this draft, we argue that there is a very small distance between supporting path migration and supporting simulatneous traffic on multipath. Instead of using an implicit algorithm to discard unused paths, we propose a simple option to allow explicit management of active and inactive paths. Once paths are explicitly managed, pretty much all other requirements for multipath support can be met by appropriate algorithms for scheduling transmissions on specific paths. These algorithms can be implemented on the sender side, and do not require specific extensions, except maybe a measurement of one way delays.

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 April 23, 2021.

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 the QUIC transport [I-D.ietf-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 can, indeed SHOULD, 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 [I-D.ietf-quic-transport].

Path migration according to [I-D.ietf-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., ack eliciting 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 an explicit mechanism for replacing the implicit signalling of path migration through data transmission, by means of a new PATH_OPTION frame.

2. Path Management Requirement

Implicit path management, as specified in [I-D.ietf-quic-transport], fulfills two 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. Explicit path management will have to fulfill similar goals: provide indications to the peer regarding preferred paths for receiving data; and, signal to the peer when resource associated with previous paths can be discarded.

We cannot mandate that path management signals be always carried on the path that they affect. For example, consider a node that wants to abandon a path through a Wi-Fi network because the signal strength

is fading. It will need to signal its peer to move traffic away from that path, but there is no guarantee that a packet sent through the old fading path will be promptly received. It is much preferable to send such management signals on a different path, for example through a cellular link. But if path management signals can be sent on arbitrary paths, they need to identify the path that they manage.

Neither addresses nor connection identifiers are good identifiers of paths. Because of NAT, addresses and ports can be changed during the transfer of packets from source to destination. Multiple connection identifiers can be used over the lifetime of a path, and there is also no strict requirement that a given connection identifier be used

on a single path. On the other hand, one can imagine an inband "path identification" frame that establishes an identifier for a specific path. That identifier can then be used in other path management frames.

The path management frames provides directions on how to use existing paths. We could imagine a number of variations, but the simplest one would be:

- Abandon a path, and release the corresponding resource.
- Mark a path as "available", i.e., allow the peer to use its own logic to split traffic among available paths.
- Mark a path as "standby", i.e., suggest that no traffic should be sent on that path if another path is available.

We could also envisage marking a path as "preferred", suggesting that all traffic would be sent to that path, but the same functionality is achieved by marking only one path as available.

There might be a delay between the moment a path is validated through the challenge response mechanism, identified using path identification frames, and managed using path management frames. During that delay, the following conventions apply:

- If a path is validated but no ack-eliciting frames or path management frames are received, the path is placed in the "standby" state.

- If ack-eliciting frames are received on a path before path management frames, the path is placed in the "available" state.

3. Path Management Extension

The path management extension is negotiated by means of the "enable_path_management" transport parameter. When support is negotiated, the peers can send or receive the PATH_IDENTIFICATION and PATH_STATUS frames.

3.1. Path Management Support Transport Parameter

The use of the PATH_IDENTIFICATION and PATH_STATUS transport frame extension is negotiated using a transport parameter:

- "enable_path_management" (TBD)

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

1. I would like to send PATH_IDENTIFICATION and PATH_STATUS frames

2. I am able to process PATH_IDENTIFICATION and PATH_STATUS frames

3. I am able to process PATH_IDENTIFICATION and PATH_STATUS frames and I would like to send them.

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

A peer that advertises its capability of sending PATH_IDENTIFICATION and PATH_STATUS frames using option values 1 or 3 MUST NOT send these frames if the other peer does not announce advertise its ability to process them by sending the enable_path_management TP with option 1 or 3.

3.2. PATH_IDENTIFICATION Frame (TBD)

The PATH_IDENTIFICATION frames carry a single identifier, the path identifier. " PATH_IDENTIFICATION Frame { Path Identifier (i) } " The frame carries the identifier of the path over which it is received, as defined by the peer sending data on that path. The identifier is a positive integer lower than 2^62.

PATH_IDENTIFICATION frames are ACK-eliciting.

Due to delays and retransmissions, several PATH_IDENTIFICATION frames may be received on the same path. In that case, the identifier with the highest value is retrained for the path. For example, if a node receives on the same path the identifiers 11, 17 and 13, the path will be identified by the number 17.

If both peers have successfully negotiated the capability of sending path management frames, each peer will send its own identifier over the path.

PATH_MANAGEMENT frames MUST only be sent in 1-RTT packets. Receiving such frames in Initial, Handshake of 0-RTT packets is a transport violation.

3.3. PATH_STATUS Frame (TBD)

The PATH_STATUS frames carry three parameters: the identifier of the path being managed, the new status of the path, and a management sequence number.

" PATH_STATUS Frame { Path Identifier (i), Path Status (i), Management Sequence Number (i) } "

The path identifier SHOULD match the value sent in a previous PATH_IDENTIFICATION frame for an existing frame. If no such value is found, the PATH_STATUS frame will be discarded.

NAT rebinding and retransmissions could cause the same identifier to be received over different paths. If that happens, PATH_MANAGEMENT

frames will apply to all the paths sharing that identifier.

PATH_STATUS frames are ACK-eliciting.

Due to delays and retransmissions, PATH_STATUS frames may be received out of order. If the Management Sequence Number is not greater than the previously received values for that path, the PATH_STATUS frame will be discarded.

If the frame is accepted, the status of the path is set to the received Path Status, for which the authorized values are:

0- Abandon 1- Standby 2- Available

Receiving any other value is an error, which SHOULD trigger the closure of the connection with a reason code PROTOCOL VIOLATION.

PATH_MANAGEMENT frames MUST only be sent in 1-RTT packets. Receiving such frames in Initial, Handshake of 0-RTT packets is a transport violation.

4. Sender side algorithms

In this minimal design, the sender is responsible for scheduling packets transmission on different paths, preferably on "Available" paths, but possibly on "StandBy" paths if all available paths are deemed unusable.

4.1. Packet Numbers and Acknowledgements

The packet number space does not depend on the path on which the packet is sent. ACK frames report the numbers of packets that have

been received so far, regardless of the path on which they have been received.

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. Senders that want to control this issue may do so by dedicating sub-ranges of packet numbers to different paths. We

expect that experience on how to manage such ranges will quickly emerge.

4.2. Congestion Control and Error Recovery

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 [I-D.ietf-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.

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. It is unclear whether this is a problem in practice, but this could motivate the use of time stamps [I-D.huitema-quic-ts] in conjunction with acknowledgements.

5. Security Considerations

TBD. There are probably ways to abuse this.

6. IANA Considerations

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

Value: TBD (using value 0xe9a in early deployments)

Parameter Name: enable_path_management

Specification: Indicates support of the Path Management options.

This document also registers 2 new values in the QUIC Frame Type registry:

Value: TBD (using value 0x9a1 in early deployments)

Frame Name: PATH_IDENTIFICATION

Specification: Identification of the path over which a packet is sent.

Value: TBD (using value 0x9a5 in early deployments)

Frame Name: PATH_STATUS

Specification: Identification of the path over which a packet is sent.

7. Acknowledgements

8. Normative References

[I-D.huitema-quic-ts]
Huitema, C., "Quic Timestamps For Measuring One-Way Delays", draft-huitema-quic-ts-03 (work in progress), August 2020.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-31 (work in progress), September 2020.

Author's Address

Christian Huitema
Private Octopus Inc.
427 Golfcourse Rd
Friday Harbor WA 98250
U.S.A

Email: huitema@huitema.net