Stream: Internet Engineering Task Force (IETF)
RFC: 10069
Category: Standards Track
Published: November 2026
ISSN: 2070-1721
Authors:
刘彦梅 (Y. Liu), Ed. *Alibaba Inc.*
马云飞 (Y. Ma) *Harvey AI*
Q. De Coninck, Ed. *University of Mons (UMONS)*
O. Bonaventure *UCLouvain and WELRI*
C. Huitema *Private Octopus Inc.*
M. Kühlewind, Ed. *Ericsson*

# RFC 10069
# Managing Multiple Paths for a QUIC Connection

## Abstract

This document specifies a multipath extension for the QUIC protocol to enable the simultaneous usage of multiple paths for a single connection. It introduces explicit path identifiers to create, delete, and manage multiple paths. This document does not specify address discovery or management, nor how applications using QUIC schedule traffic over multiple paths.

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc10069.

## Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions

with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction 4
1.1. Conventions and Definitions 5
2. Connection Lifecycle and Packet Protection 5
2.1. initial_max_path_id Transport Parameter 5
2.2. Relation to Other Transport Parameters 6
2.3. Handling ACK and PATH_ACK in 0-RTT and 1-RTT 7
2.4. Nonce Calculation after Handshake Completion 7
2.5. Key Phase Update Process 8
2.6. Connection Closure 9
3. Path Management 9
3.1. Path Initiation and Validation 9
3.1.1. Path Establishment Example 10
3.1.2. Relation to Probing and Migration 11
3.1.3. Address Validation Token 12
3.2. Handling Connection IDs 12
3.2.1. Issuing New Connection IDs 12
3.2.2. Rotating and Retiring Connection IDs 13
3.3. Path Status Management 13
3.4. Path Close 14
3.4.1. Path Closure Example 15
3.4.2. Avoiding Spurious Stateless Resets 16
3.4.3. Handling PATH_ACK for Abandoned Paths 16
4. New Frames 17
4.1. PATH_ACK Frame 17
4.2. PATH_ABANDON Frame 17
4.2.1. Error Codes 18

4.3. PATH_STATUS_AVAILABLE and PATH_STATUS_BACKUP Frames 19
4.4. PATH_NEW_CONNECTION_ID Frame 20
4.5. PATH_RETIRE_CONNECTION_ID Frame 21
4.6. MAX_PATH_ID Frame 22
4.7. PATHS_BLOCKED and PATH_CIDS_BLOCKED Frames 22
5. Implementation Considerations 24
5.1. Connection ID Changes, Migration, and NAT Rebindings 24
5.2. Using Multiple Paths on the Same 4-tuple 24
5.3. Congestion Control 25
5.4. Computing Path RTT 25
5.5. Packet Scheduling 27
5.6. Retransmissions 27
5.7. PTO Expiration 27
5.8. Paths Having Different PMTU Sizes 28
5.9. Idle Timeout and Keep-Alives 28
6. IANA Considerations 28
7. Security Considerations 30
7.1. Memory Allocation for Per-Path Resources 30
7.2. Denial of Service with Multiple Paths 31
7.3. Cryptographic Handshake and AEAD Nonce 31
8. References 31
8.1. Normative References 31
8.2. Informative References 32
Acknowledgments 32
Authors' Addresses 33

# 1. Introduction

This document specifies an extension to QUIC version 1 [QUIC-TRANSPORT] to enable the simultaneous usage of multiple paths for a single connection, using the same or different 4-tuples (of source/destination port numbers and source/destination IP addresses).

Connection migration as specified in Section 9 of [QUIC-TRANSPORT] directs a peer to switch sending through a new preferred path and, if successful, to release resources associated with the old path. The multipath extension specified in this document builds on this mechanism but introduces a path identifier, or path ID, to manage connection IDs and packet number spaces per path, enabling the use of multiple paths simultaneously. As such, a path in this extension is defined by its path ID. Note that it therefore is possible to have multiple paths / path IDs for the same 4-tuple.

The connection ID of a packet binds the packet to a path ID and therefore to a packet number space. That means each connection ID is associated with exactly one path ID but multiple connection IDs are usually issued for each path ID. The same path ID is used in both directions, starting with 0 for the initial path. Path IDs are monotonically increasing and cannot be reused.

This extension uses multiple packet number spaces, specifically one for each path. Each path ID-specific packet number space starts at packet number 0. As such, each path maintains distinct packet number states for sending and receiving packets, as in [QUIC-TRANSPORT]. Using multiple packet number spaces enables direct use of the loss detection and congestion control mechanisms defined in [QUIC-RECOVERY] on a per-path basis. However, use of multiple packet number spaces requires non-zero connection IDs in order to identify the path and the respective packet number space, as well as a modified Authenticated Encryption with Associated Data (AEAD) calculation including the path ID (see Section 2.4).

As such, this extension specifies a departure from the specification of path management in Section 9 of [QUIC-TRANSPORT] and therefore requires a new transport parameter, as specified in Section 2.1, to indicate support of the multipath extension specified in this document.

Further, this document specifies the needed path management mechanisms for path initiation in Section 3.1, handling of per-path connection IDs in Section 3.2, signaling of preferred path usage in Section 3.3, and explicit removal of paths that have been abandoned in Section 3.4. Note that in this extension, a QUIC server does not initiate the creation of a path, but it has to validate a new path created by a client.

This extension does not cover address discovery and management. The extension specified in this document can be used as is if address management and the actual decision to set up or tear down paths are handled by the application. This document does not prevent future extensions from defining mechanisms to cope with the remaining scenarios.

Further, this document does not specify scheduling algorithms that define how multiple, simultaneously open paths are used to send packets. As these differ depending on application requirements, only some basic implementation guidance is discussed in Section 5. This extension

can be used with different scheduling algorithms that, e.g., can range from support for failover to simultaneous use of the aggregated capacity across all open paths. There are currently no IETF specifications that define scheduling algorithms for simultaneously (i.e., concurrently) using multiple paths.

Specifically, while failover between Wi-Fi and mobile networks is a well-known multipath use case, it only temporarily uses two paths at the same time to avoid transmission pauses. Simultaneous path usage generally, however, needs more consideration than specified in this document to avoid negative performance impacts, e.g., when stream data is distributed over multiple paths with different delays.

The operational considerations for QUIC are addressed in [RFC9312]. They also apply to QUIC connections using the extensions defined in this document. An additional complexity is that applications might use a combination of monitored and non-monitored paths, but that complexity already exists when using path migration as defined in [QUIC-TRANSPORT].

### 1.1. 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.

This document uses the terminology defined in [QUIC-TRANSPORT]. When this document uses the term "path", it refers to the notion of "network path" used in [QUIC-TRANSPORT].

The packet diagrams in this document use the conventions defined in Section 1.3 of [QUIC-TRANSPORT], including the notation (i) to denote variable-length integers, encoded as specified in Section 16 of [QUIC-TRANSPORT].

## 2. Connection Lifecycle and Packet Protection

This document defines a new transport parameter initial_max_path_id to indicate the support of the multipath extension. If either of the endpoints does not advertise the initial_max_path_id transport parameter, then both endpoints **MUST NOT** use any frame or mechanism defined in this document. If both endpoints advertise the initial_max_path_id transport parameter, once the handshake is completed, a new AEAD usage applies to all 1-RTT packets, as specified in Section 2.4, and new paths can be used, as specified in Section 3. As specified in Section 4, the new frames defined in this document can only be sent in 1-RTT packets.

The transport parameter negotiation enables incremental deployment of implementations of the multipath extension.

### 2.1. initial_max_path_id Transport Parameter

The new transport parameter is defined as follows:

initial_max_path_id (parameter ID 0x3e): Parameter value is a variable-length integer specifying the maximum path ID an endpoint is willing to maintain at connection initiation. This value **MUST NOT** exceed $2^{32}$-1, which is the maximum allowed value for the path ID due to restrictions on the nonce calculation (see Section 2.4).

The initial_max_path_id transport parameter limits the initial maximum number of open paths that can be used during a connection. For example, if initial_max_path_id is set to 1, only connection IDs associated with path IDs 0 and 1 should be issued by the peer. If an endpoint receives an initial_max_path_id transport parameter with value 0, the peer aims to enable the multipath extension without allowing extra paths immediately.

Setting initial_max_path_id parameter is equivalent to sending a MAX_PATH_ID frame (Section 4.6) with the same value. As such to allow for the use of more paths later, endpoints can send the MAX_PATH_ID frame to increase the maximum allowed path ID.

If an initial_max_path_id transport parameter value higher than $2^{32}$-1 is received, the receiver **MUST** close the connection with an error of type TRANSPORT_PARAMETER_ERROR.

When advertising the initial_max_path_id transport parameter, endpoints **MUST** use Source and Destination Connection IDs with non-zero lengths. If an initial_max_path_id transport parameter is received and the carrying packet contains a zero-length connection ID, the receiver **MUST** treat this as a connection error of type PROTOCOL_VIOLATION and close the connection.

Cipher suites with a nonce shorter than 12 bytes cannot be used together with the multipath extension. If such a cipher suite is selected and the use of the multipath extension is supported, endpoints **MUST** abort the handshake with an error of type TRANSPORT_PARAMETER_ERROR.

The initial_max_path_id parameter **MUST NOT** be remembered for use in a subsequent connection (Section 7.4.1 of [QUIC-TRANSPORT]).

## 2.2. Relation to Other Transport Parameters

When the QUIC multipath extension is used, the active_connection_id_limit transport parameter [QUIC-TRANSPORT] limits the maximum number of active connection IDs per path. As defined in Section 5.1.1 of [QUIC-TRANSPORT], connection IDs that are issued and not retired are considered active.

If an endpoint receives a disable_active_migration transport parameter, it is forbidden to establish new paths to the peer's handshake address. However, establishment of additional paths to other peer addresses (e.g., carried by peer's preferred_address) is immediately valid.

If the server uses the preferred_address transport parameter, clients cannot assume that the initial server address and the addresses contained in this parameter can be simultaneously used for multipath (Section 9.6.2 of [QUIC-TRANSPORT]). Use of the preferred address with the same local address is considered as a migration event that does not change the path ID. A such, the path ID for the connection ID specified in the preferred_address transport parameter is 0.

## 2.3. Handling ACK and PATH_ACK in 0-RTT and 1-RTT

The PATH_ACK frame (see Section 4.1) is used to acknowledge 1-RTT packets. Compared to the ACK frame, as specified in Section 19.3 of [QUIC-TRANSPORT], the PATH_ACK frame additionally contains the path ID to identify the path-specific packet number space. ACK frames when used with the multipath extension acknowledge packets for the path with path ID 0. As multipath support is unknown during the handshake, acknowledgments of Initial and Handshake packets are sent using ACK frames.

After the handshake concludes with support for the multipath extension, endpoints **SHOULD** use PATH_ACK frames instead of ACK frames, including for so far unacknowledged 0-RTT packets using path ID 0. Endpoints **MUST** still process ACK frames that acknowledge 0-RTT packets or 1-RTT packets. For example, a sender could negotiate multipath support for later use and keep only the initial path with path ID 0 for a while. During this single-path period, the sender might prefer to send ACK frames.

## 2.4. Nonce Calculation after Handshake Completion

Section 5.3 of [QUIC-TLS] specifies AEAD usage, particularly the use of a nonce, N, formed by combining the packet protection Initialization Vector (IV) with the packet number. When multiple packet number spaces are used, the packet number alone would not guarantee the uniqueness of the nonce. Therefore, the nonce N is calculated for 1-RTT packets if the multipath extension is used by combining the packet protection IV with the packet number and with the 32 bits of the path ID. In order to guarantee the uniqueness of the nonce, the path ID is limited to a max value of $2^{32}$-1, as specified in Section 2.1.

To calculate the nonce, a 96-bit Path-and-Packet-Number (PPN) is composed of the 32 bits of the path ID in network byte order, two zero bits, and the 62 bits of the reconstructed QUIC packet number in network byte order, as illustrated in Figure 1.

```
PPN {
  Path ID (32),
  Zeroes (2) = 0b00,
  Packet Number (62)
}
```

*Figure 1: 96 Bits Path-And-Packet-Number*

The IV length is equal to the nonce length. If the IV is larger than 96 bits, the path-and-packet-number is left-padded with zeros to the size of the IV. The exclusive OR of the padded packet number and the IV forms the AEAD nonce. An AEAD algorithm where the nonce length is less than 12 bytes cannot be used with the QUIC multipath extension. The following figure illustrates this for a 96-bit IV.

```
IV(12);
N(12) = IV xor PPN;
```

*Figure 2: Nonce Calculation*

For example, assuming the IV value is 0x6b26114b9cba2b63a9e8dd4f, the path ID is 3, and the packet number is 54321 (hex value 0xd431), the nonce will be set to 0x6b2611489cba2b63a9e8097e, as illustrated below (where ⊕ represents exclusive OR):

```
   IV:      6b26114b9cba2b63a9e8dd4f
⊕  PPN:     00000003000000000000d431
------------------------------------
   Nonce:   6b2611489cba2b63a9e8097e
```

*Figure 3: Example Nonce Calculation*

## 2.5. Key Phase Update Process

The Key Phase bit update process is specified in Section 6 of [QUIC-TLS]. The general principles of key update are not changed in this specification. Following [QUIC-TLS], the Key Phase bit is used to indicate which packet protection keys are used to protect the packet. The Key Phase bit is toggled to signal each subsequent key update.

Because of network delays, packets protected with the older key might arrive later than the packets protected with the new key. However, receivers can solely rely on the Key Phase bit to determine the corresponding packet protection key, assuming that there is a sufficiently large interval between two consecutive key updates (Section 6.5 of [QUIC-TLS]).

When this specification is used, endpoints **SHOULD** wait for at least three times the largest Probe Timeout (PTO) (see Section 6.2 of [QUIC-RECOVERY]) among all the paths before initiating a new key update after receiving an acknowledgment that confirms the receipt of the previous key update. This interval is different from that in [QUIC-TLS], which used three times the PTO of the single path.

Since packets that arrive after their decryption key has been discarded will be dropped, the choice of three times the largest PTO is a trade-off: Longer delays reduce the probability of losing packets, but keeping old keys longer can negatively impact the security of the protocol. The use of three times the largest PTO aims to minimize packet lost for all paths and therefore limits the impact on performance.

Following Section 5.4 of [QUIC-TLS], the Key Phase bit is protected, so sending multiple packets with the Key Phase bit flipping at the same time should not cause activity across different paths to be linkable by an observer.

### 2.6. Connection Closure

CONNECTION_CLOSE frames and their processing are unchanged from [QUIC-TRANSPORT]. They can be sent on any open path. Section 10.2 of [QUIC-TRANSPORT] specifies that the closing and draining connection states "**SHOULD** persist for at least three times the current PTO". When this specification is used, these states **SHOULD** instead persist for at least three times the largest PTO among all paths.

## 3. Path Management

After completing the handshake indicating multipath support, endpoints can start using multiple paths. In accordance with Section 9 of [QUIC-TRANSPORT], this extension only enables clients to open a new path. The client can open a new path when both endpoints have issued available connection IDs for at least one unused, common path ID, as the same path ID is used in both directions.

This document specifies path initiation (see Section 3.1), issuing and retirement of per-path connection IDs (see Section 3.2), path status management (see Section 3.3), and path closure (see Section 3.4). However, this document does not specify when a client decides to initiate or close a path or how multiple open paths are used for sending.

For path management, this extension specifies the following frames in Section 4:

- PATH_ABANDON (see Section 4.2)
- PATH_STATUS_BACKUP (see Section 4.3)
- PATH_STATUS_AVAILABLE (see Section 4.3)
- PATH_NEW_CONNECTION_ID (see Section 4.4)
- PATH_RETIRE_CONNECTION_ID (see Section 4.5)
- MAX_PATH_ID (see Section 4.6)
- PATHS_BLOCKED (see Section 4.7)
- PATH_CIDS_BLOCKED (see Section 4.7)

### 3.1. Path Initiation and Validation

To open a new path, an endpoint **MUST** use a new connection ID associated with an unused path ID. An endpoint **MUST** use a connection ID associated to the same path ID as used in the packet received by the endpoint when it intends to send packets on the same path.

A client that wants to use a new path **MUST** validate the peer's address as described in Section 8.2 of [QUIC-TRANSPORT], unless it has previously validated the 4-tuple used for that path.

After receiving packets from the client on a new path, if the server decides to use the new path, the server **MUST** validate the peer's address before sending any data as described in (Section 8.2 of [QUIC-TRANSPORT]), unless it has previously validated the 4-tuple used for that path. Until the client's address is validated, the anti-amplification limit from Section 8 of [QUIC-TRANSPORT] applies.

If an endpoint sends a PATH_RESPONSE (Section 19.18 of [QUIC-TRANSPORT]), it **MUST** be sent on the same path as used by the packet that contained the PATH_CHALLENGE frame (Section 19.17 of [QUIC-TRANSPORT]) using a connection ID associated with the same path ID.

The server might receive packets for a yet unused path ID that do not contain a PATH_CHALLENGE frame. Such packets are valid if they can be properly decrypted given a valid connection ID.

Each endpoint **MUST** also validate that a minimum QUIC packet MTU of 1200 bytes is supported on the path. This can be done during initial path validation or separately later if the amplification limit prevents it initially, as specified in Section 8.2.1 of [QUIC-TRANSPORT].

An endpoint that receives packets on a new path and does not want to establish this path is expected to close the path by sending a PATH_ABANDON on another path, as specified in Section 3.4.

An endpoint that has no active connection ID for this path or lacks other resources to immediately configure a new path could delay sending the PATH_RESPONSE until sufficient resources are available. Long delays might cause the peer to repeat the PATH_CHALLENGE and eventually send a PATH_ABANDON, in which case the procedures specified in Section 3.4 apply.

PATH_ACK frames (see Section 4.1) can be returned on any path. If the PATH_ACK is preferred to be sent on the same path as the acknowledged packet (see Section 5.4 for further guidance), it can be beneficial to bundle a PATH_ACK frame with the PATH_RESPONSE frame during path validation.

If validation succeeds, the client can continue to use the path. If validation fails, the client **MUST NOT** use the path and can remove any status associated to the path initiation attempt. As the used path ID is consumed either way, the endpoint **MUST** explicitly close the path, as specified in Section 3.4.

#### 3.1.1. Path Establishment Example

In the example below, it is assumed that both endpoints have indicated an initial_max_path_id value of at least 2, which means both endpoints can use path IDs 0, 1, and 2. Note that path ID 0 is already used for the initial path.

```
Client                                                  Server

(Provide new CIDs for path 1 on an existing path 0)
1-RTT[X]: DCID=S0, PATH_NEW_CONNECTION_ID[C1, Seq=0, PathID=1] -->
        <-- 1-RTT[Y]: DCID=C0,
                      PATH_NEW_CONNECTION_ID[S1, Seq=0, PathID=1],
                      PATH_ACK[PathID=0, PN=X]
        <-- 1-RTT[Y+1]: DCID=C0, PATH_NEW_CONNECTION_ID[S2, Seq=0,
                                                        PathID=2]
...
(start sending packets on a new path using path ID 1)
1-RTT[0]: DCID=S1, PATH_CHALLENGE[X] -->
     <-- 1-RTT[0]: DCID=C1, PATH_RESPONSE[X], PATH_CHALLENGE[Y],
                                          PATH_ACK[PathID=1, PN=0]
1-RTT[1]: DCID=S1, PATH_RESPONSE[Y],
         PATH_ACK[PathID=1, PN=0], ... -->
```

*Figure 4: Example of New Path Establishment*

In Figure 4, the endpoints first exchange new available connection IDs with the PATH_NEW_CONNECTION_ID frame, as further explained in Section 3.2. In this example, the client provides one connection ID (C1 with path ID 1), and the server provides two connection IDs (S1 with path ID 1 and S2 with path ID 2).

Before the client opens a new path by sending a packet on that path with a PATH_CHALLENGE frame, it has to check whether there is an unused connection ID for the same unused path ID available for each side. In this example, the path ID 1 is used, which is the smallest unused path ID available as recommended in Section 3.2. Respectively, the client chooses the connection ID S1 as the Destination Connection ID of the new path when sending the PATH_CHALLENGE frame. The server replies with a PATH_RESPONSE bundled with the PATH_ACK using connection ID C1 associated with the same path ID.

#### 3.1.2. Relation to Probing and Migration

Section 9.1 of [QUIC-TRANSPORT] introduces the concept of "probing" and "non-probing" frames. A packet that contains at least one "non-probing" frame is a "non-probing" packet. Migration, as specified in Section 9.2 of [QUIC-TRANSPORT], is initiated by sending packets containing non-probing frames on a new (validated) path; however, it uses the same path ID as on the old path. When the multipath extension is negotiated, the reception of any packet (no matter if "probing" or "non-probing") on a new path with a new, so far unused path ID does not impact the path status of any existing path. Therefore, any frame can be sent on a new path with a new path ID at any time as long as the anti-amplification limits (see Section 21.1.1.1 of [QUIC-TRANSPORT]) and the congestion control limits for this path are respected.

An endpoint could receive a packet with a connection ID associated to an active path ID where the packet's 4-tuple does not match the 4-tuple currently used with that path ID. This **MUST** be treated as path migration, as specified in Section 9.3 of [QUIC-TRANSPORT], with the constraint that all connection IDs used during path migration **MUST** be associated with the current path ID of the path being migrated.

#### 3.1.3. Address Validation Token

As specified in Section 9.3 of [QUIC-TRANSPORT], the server is expected to send a new address validation token to a client following the successful validation of a new client address. The client will receive several tokens. When considering using a token for subsequent connections, it might be difficult for the client to pick the "right" token among multiple tokens obtained in a previous connection. The client is likely to fall back to the strategy specified in Section 8.1.3 of [QUIC-TRANSPORT], i.e., picking the last received token. To avoid issues when clients make the "wrong" choice, a server **SHOULD** issue tokens that are capable of validating any of the previously validated addresses. Including more addresses increases the probability that the token will be useful in the future but at the cost of a larger token. Further guidance on token usage can be found in Section 8.1.3 of [QUIC-TRANSPORT].

### 3.2. Handling Connection IDs

When the multipath extension is used, endpoints have to use the PATH_NEW_CONNECTION_ID and PATH_RETIRE_CONNECTION_ID frames to indicate the respective path ID together with the associated sequence number (see Section 5.1.1 of [QUIC-TRANSPORT]), at least for all paths with a path ID other than 0. Each path ID has its own connection ID sequence number space whose initial value is 0.

Endpoints **SHOULD** also use PATH_NEW_CONNECTION_ID and PATH_RETIRE_CONNECTION_ID for the initial path with path ID 0. However, the use of NEW_CONNECTION_ID and RETIRE_CONNECTION_ID is still valid and endpoints need to process these frames as corresponding to path ID 0.

#### 3.2.1. Issuing New Connection IDs

In order to let the peer open new paths, it is **RECOMMENDED** to proactively issue at least one Connection ID for each unused path ID up to the minimum of the peer's and the local maximum path ID limits.

If an endpoint does not want to issue connection IDs for all unused path IDs for any reason, it **SHOULD NOT** introduce discontinuity in the issuing of path IDs, as path initiation requires available connection IDs for the same path ID on both sides. For instance, if the maximum path ID limit is 2 and the endpoint wants to provide connection IDs for only one path ID inside range [1, 2], it should select path ID 1 (and not path ID 2).

Similarly, endpoints **SHOULD** consume path IDs in a continuous way, i.e., when creating paths. However, endpoints cannot expect to receive new connection IDs or path initiation attempts with in-order use of path IDs due to out-of-order delivery or path validation failure.

Each endpoint maintains the set of connection IDs received from its peer for each path, any of which it can use when sending packets on that path; see also Section 5.1 of [QUIC-TRANSPORT]. Usually, it is desired to provide at least one additional connection ID for all used paths to allow for (unintentional) migration events (Section 9.5 of [QUIC-TRANSPORT]).

As further specified in Section 5.1 of [QUIC-TRANSPORT], connection IDs cannot be issued more than once on the same connection and therefore are unique for the scope of the connection, regardless of the associated path ID.

Endpoints **MUST NOT** issue new connection IDs with path IDs greater than the Maximum Path Identifier field in MAX_PATH_ID frames (see Section 4.6) or the value of initial_max_path_id transport parameter if no MAX_PATH_ID frame was received yet. Receipt of a frame with a greater path ID is a connection error as specified in Section 4.

When an endpoint cannot open a path because there are no unused path IDs, it **SHOULD** either send a MAX_PATH_ID frame to increase the maximum path ID limit (when limited by the sender) or a PATHS_BLOCKED frame (see Section 4.7) to inform the peer that its current limit prevented the creation of the new path.

#### 3.2.2. Rotating and Retiring Connection IDs

Section 5.1.2 of [QUIC-TRANSPORT] indicates that an endpoint can change the connection ID it uses to another available one at any time during the connection. For the extension specified in this document, endpoints **MUST** only rotate to another connection ID associated with the same path ID. Use of a connection ID associated with another path ID will be considered as an attempt to open a new path instead.

An endpoint is supposed to retire any connection ID that is not being used, and the server is expected to provide replacements, as specified in Section 5.1.2 of [QUIC-TRANSPORT]. As such, when receiving a PATH_RETIRE_CONNECTION_ID frame, an endpoint **SHOULD** provide new connection IDs for that path, if still open, using PATH_NEW_CONNECTION_ID frames.

While it is expected that the peer provides at least one unused connection ID for all active paths using the PATH_NEW_CONNECTION_ID after retirement of an old connection ID, an endpoint **MAY** send a PATH_CIDS_BLOCKED (see Section 4.7) if it wants to change the connection ID but no unused connection ID for a that path is available. Further, an endpoint **MAY** also send a PATH_CIDS_BLOCKED frame if it wants to open a new path and has no connection IDs available for an unused path ID, even though the Maximum Path Identifier value would allow for more paths.

Retirement of connection IDs will not retire the path ID that corresponds to the connection ID or any other path resources as the packet number space is associated to the path ID.

The peer that sends the PATH_RETIRE_CONNECTION_ID frame can keep sending data on the path that the retired connection ID was used on but has to use a different connection ID for the same path ID when doing so.

### 3.3. Path Status Management

An endpoint can send PATH_STATUS_BACKUP and PATH_STATUS_AVAILABLE frames (see Section 4.3) to inform the peer that it should send packets on the paths with the preference expressed by these frames. Note that an endpoint might not follow the peer's advertisements, but these frames are still a clear signal of the peer's preference of path usage.

Each peer indicates its preference of path usage independently of the other peer. That means that peers could have different usage preferences for the same path. Depending on the data sender's decisions, this might lead to usage of paths that have been indicated as "backup" by the peer or non-usage of some locally available paths.

PATH_STATUS_AVAILABLE indicates that a path is "available", i.e., it suggests to the peer to use its own logic to split traffic among available paths.

PATH_STATUS_BACKUP suggests that a path should only be used as backup, i.e., that no traffic should be sent on that path if another path is available and usable. If all established paths are indicated as backup paths, no guidance is provided about which path should be used.

Similarly, if no frame indicating a path usage preference was received for a certain path, the preference of the peer is unknown and the sender needs to decide based on its own local logic if the path should be used.

If an endpoint starts using a backup path because it has detected issues on the paths marked as "available", it is **RECOMMENDED** to update its own path state signaling such that the peer avoids using the broken path. An endpoint that detects a path breakage can also explicitly close the path by sending a PATH_ABANDON frame (see Section 3.4) in order to avoid that its peer keeps using it and enable faster switchover to a backup path. If the endpoints do not want to close the path immediately, as connectivity could be re-established, PING frames can potentially be used to quickly detect connectivity changes and switch back in a timely way.

The PATH_STATUS_AVAILABLE and PATH_STATUS_BACKUP frames share a common, per-path sequence number space to detect and ignore outdated information, as further described in Section 4.3. This is needed as they might arrive out of order, e.g., if sent using different paths.

## 3.4. Path Close

At any time in the connection, each endpoint can decide to abandon a path, for example, following changes in local connectivity or local preferences. An endpoint that wants to abandon a path **MUST** explicitly close the path by sending a PATH_ABANDON frame (see Section 4.2). This is true whether the decision to close the path results from implicit signals such as an idle timeout (see Section 5.9) or packet losses, as well as for any other reason such as management of local resources.

Endpoints that send a PATH_ABANDON frame **MUST** treat all connection IDs received from the peer for the path ID indicated in the PATH_ABANDON as immediately retired and subsequently cannot send any packet on that path anymore. Note that while abandoning a path will cause connection ID retirement, the inverse is not true: Retiring the associated connection IDs does not indicate path abandonment (see further Section 3.2).

PATH_ABANDON frames can be sent on any open path, not only on the path that is intended to be closed. It is **RECOMMENDED** to send the PATH_ABANDON frames on another open path, especially if connectivity on the to-be-abandoned path is expected to be broken.

When an endpoint receives a PATH_ABANDON frame, it **MUST** send a corresponding PATH_ABANDON frame if it has not already done so and respectively treat all connection IDs received from the peer for that path as immediately retired. While that means retired connection IDs received from the peer cannot be used for sending anymore, packets from the peer might still be in transit. Therefore, knowledge of the connection IDs issued to the peer and of the state of the number space associated to the path **SHOULD** be retained for 3 PTOs after the PATH_ABANDON frame has been received. This avoids generating spurious Stateless Reset packets, as discussed in Section 3.4.2, and helps acknowledge any potentially reordered, outstanding packets from the peer (see Section 3.4.3).

It is also possible that an endpoint will receive a PATH_ABANDON frame before receiving or sending any traffic on a path. For example, if the client tries to initiate a path and the path cannot be established, it will send a PATH_ABANDON frame (see Section 3.1). An endpoint could also decide to abandon an unused path for any other reason, for example, removing a hole from the sequence of path IDs in use. This is not an error.

If a peer sends a PATH_ABANDON frame but never receives a corresponding PATH_ABANDON frame, it might not be able to remove path state. It is left to the implementation to handle this unexpected behavior, as it does not impact interoperability. If the endpoint is no longer willing to process the issued connection IDs for the abandoned path, it **MAY** close the connection but **SHOULD** wait at least 3 PTOs after sending the PATH_ABANDON frame.

After a path is abandoned, the path ID **MUST NOT** be reused for new paths, as the path ID is part of the nonce calculation (see Section 2.4).

If a PATH_ABANDON frame is received for the only open path of a QUIC connection, the receiving peer **SHOULD** send a CONNECTION_CLOSE frame and enter the closing state. Alternatively, a client **MAY** instead try to open a new path, if available, and only initiate connection closure if path validation fails or a CONNECTION_CLOSE frame is received from the server. Similarly, the server **MAY** wait for a short, limited time, such as one PTO, to see if a packet is received on a new path before sending the CONNECTION_CLOSE frame.

Note that other explicit closing mechanisms of [QUIC-TRANSPORT] still apply on the whole connection. In particular, the reception of either a CONNECTION_CLOSE (Section 10.2 of [QUIC-TRANSPORT]) or a Stateless Reset (Section 10.3 of [QUIC-TRANSPORT]) closes the connection.

### 3.4.1. Path Closure Example

In the example below, the client wants to close the path with path ID 0. It sends the PATH_ABANDON frame to terminate the path with path ID 0 on the path with path ID 1 using the connection ID S1. After receiving the PATH_ABANDON frame for path ID 0, the server also sends a PATH_ABANDON frame with path ID 0 together with a PATH_ACK frame on the same path using connection ID C1.

```
Client                                                    Server

(client tells server to abandon a path with path ID 0)
1-RTT[X]: DCID=S1 PATH_ABANDON[path ID=0]->
                          (server tells client to abandon a path)
                    <-1-RTT[Y]: DCID=C1 PATH_ABANDON[path ID=0],
                                          PATH_ACK[PATH ID=1, PN=X]
1-RTT[U]: DCID=S1 PATH_ACK[path ID=1, PN=Y] ->
```

*Figure 5: Example of Closing a Path*

Note that if the PATH_ABANDON frame is instead sent on the to-be-abandoned path, the last acknowledgment still needs to be sent on a different path as no further packets can be sent on the abandoned path after the PATH_ABANDON frame.

#### 3.4.2. Avoiding Spurious Stateless Resets

Due to network delays, packets sent on an abandoned path can arrive well after the connection IDs have been retired. If not recognized as bound to the local connection, such a packet triggers the peer to send a Stateless Reset packet. The requirement to retain knowledge of connection ID and about the packet number space for 3 PTOs after receiving a PATH_ABANDON frame, as specified in Section 3.4 above, is intended to reduce the risk of sending such spurious stateless packets, but it cannot completely avoid that risk.

Section 10.3 of [QUIC-TRANSPORT] specified that the Stateless Reset Tokens associated with retired connection IDs cannot be used to identify Stateless Reset packets. The immediate retirement of connection IDs received from the peer for an abandoned path guarantees that spurious Stateless Reset packets sent by the peer will not cause the closure of the QUIC connection.

#### 3.4.3. Handling PATH_ACK for Abandoned Paths

When an endpoint sends a PATH_ABANDON frame, there might still be some packets in transit from the peer. Further, if an endpoint receives a PATH_ABANDON frame, it might still receive reordered packets on the abandoned path. Endpoints **SHOULD** promptly send PATH_ACK frames for all unacknowledged packets received on an abandoned path if path state is still retained to do so.

PATH_ACK frames have to be sent on a different path than the path being abandoned after sending the PATH_ABANDON frame as connection IDs are immediately retired.

When an endpoint finally deletes all state associated with the path, the packets sent over the path and not yet acknowledged **MUST** be considered lost. PATH_ACK frames received with an abandoned path ID are silently ignored, as specified in Section 4.

# 4. New Frames

All frames defined in this document **MUST** only be sent in 1-RTT packets. If an endpoint receives a multipath-specific frame in a different packet type, it **MUST** close the connection with an error of type PROTOCOL_VIOLATION.

Receipt of multipath-specific frames that use a path ID that is greater than the announced Maximum Paths value in the MAX_PATH_ID frame or in the initial_max_path_id transport parameter, if no MAX_PATH_ID frame was received yet, **MUST** be treated as a connection error of type PROTOCOL_VIOLATION.

If an endpoint receives a multipath-specific frame with a path ID that it cannot process anymore (e.g., because the path might have been abandoned), it **MUST** silently ignore the frame.

## 4.1. PATH_ACK Frame

The PATH_ACK frame (types 0x3e and 0x3f) is an extension of the ACK frame specified in Section 19.3 of [QUIC-TRANSPORT]. It is used to acknowledge packets that were sent on different paths, as each path has its own packet number space. If the frame type is 0x3f, PATH_ACK frames also contain the sum of QUIC packets with associated Explicit Congestion Notification (ECN) marks received on the acknowledged packet number space up to this point.

PATH_ACK frames are formatted as shown in Figure 6.

```
PATH_ACK Frame {
  Type (i) = 0x3e..0x3f,
  Path Identifier (i),
  Largest Acknowledged (i),
  ACK Delay (i),
  ACK Range Count (i),
  First ACK Range (i),
  ACK Range (..) ...,
  [ECN Counts (..)],
}
```

*Figure 6: PATH_ACK Frame Format*

Compared to the ACK frame specified in [QUIC-TRANSPORT], the following field is added:

Path Identifier: The path ID associated with the packet number space of the 0-RTT and 1-RTT packets that are acknowledged by the PATH_ACK frame.

## 4.2. PATH_ABANDON Frame

The PATH_ABANDON frame informs the peer to abandon a path. After the PATH_ABANDON frame is sent on a path, the path can no longer be used for sending.

PATH_ABANDON frames are formatted as shown in Figure 7.

```
PATH_ABANDON Frame {
  Type (i) = 0x3e75,
  Path Identifier (i),
  Error Code (i),
}
```

*Figure 7: PATH_ABANDON Frame Format*

PATH_ABANDON frames contain the following fields:

Path Identifier: The path ID associated to the to-be-abandoned path.

Error Code: A variable-length integer that indicates the reason for abandoning this path. NO_ERROR(0x0) indicates that the path is being abandoned without any error being encountered. Other error codes can be found in Section 4.2.1.

PATH_ABANDON frames are ack-eliciting. If a packet containing a PATH_ABANDON frame is considered lost, the peer **SHOULD** repeat it.

Use of the PATH_ABANDON frame is specified in Section 3.4.

#### 4.2.1. Error Codes

QUIC transport error codes are 62-bit unsigned integers (see Section 20.1 of [QUIC-TRANSPORT]). In addition to NO_ERROR(0x0), the following QUIC error codes are defined for use in the PATH_ABANDON frame:

APPLICATION_ABANDON_PATH (0x3e): The endpoint is abandoning the path at the request of the application.

PATH_RESOURCE_LIMIT_REACHED (0x3e75): The endpoint is abandoning the path because it cannot allocate sufficient resources to maintain it.

PATH_UNSTABLE_OR_POOR (0x3e76): The endpoint is abandoning the path because the used interface is observed to be unstable or performance is considered poor. This condition can occur, e.g., due to frequent handover events during high-speed mobility or due to a weak wireless signal.

NO_CID_AVAILABLE_FOR_PATH (0x3e77): The endpoint is abandoning the path due to the lack of a connection ID for this path. This might occur when the peer initiates a new path but has not provided a corresponding connection ID for the path ID (or the packet containing the connection IDs has not arrived yet).

### 4.3. PATH_STATUS_AVAILABLE and PATH_STATUS_BACKUP Frames

PATH_STATUS_AVAILABLE frames (type=0x3e77) are used by endpoints to inform the peer that the indicated path is available for sending.

PATH_STATUS_AVAILABLE frames are formatted as shown in Figure 8.

```
PATH_STATUS_AVAILABLE Frame {
  Type (i) = 0x3e77,
  Path Identifier (i),
  Path Status Sequence Number (i),
}
```

*Figure 8: PATH_STATUS_AVAILABLE Frame Format*

PATH_STATUS_BACKUP frames (type=0x3e76) are used by endpoints to inform the peer about its preference to not use the indicated path for sending.

PATH_STATUS_BACKUP frames are formatted as shown in Figure 9.

```
PATH_STATUS_BACKUP Frame {
  Type (i) = 0x3e76
  Path Identifier (i),
  Path Status Sequence Number (i),
}
```

*Figure 9: PATH_STATUS_BACKUP Frame Format*

Both PATH_STATUS_AVAILABLE and PATH_STATUS_BACKUP frames contain the following fields:

Path Identifier: The path ID that the status update corresponds to. All path IDs below the maximum path ID limit can be indicated, even if the path is not in active use yet.

Path Status Sequence Number: A variable-length integer specifying the per-path sequence number assigned for this frame.

The sequence number space is common to the two frame types, and monotonically increasing values **MUST** be used when sending PATH_STATUS_AVAILABLE or PATH_STATUS_BACKUP frames for a given path ID.

Frames might be received out of order. A peer **MUST** ignore an incoming PATH_STATUS_AVAILABLE or PATH_STATUS_BACKUP frame if it previously received another PATH_STATUS_BACKUP frame or PATH_STATUS_AVAILABLE frame for the same path ID with a Path Status Sequence Number equal to or higher than the Path Status Sequence Number of the incoming frame.

The requirement of monotonically increasing sequence numbers is per path. Receivers could very well receive the same sequence number for PATH_STATUS_AVAILABLE or PATH_STATUS_BACKUP frames on different paths. As such, the receiver of the PATH_STATUS_AVAILABLE or PATH_STATUS_BACKUP frame needs to use and compare the sequence numbers separately for each path ID.

PATH_STATUS_BACKUP and PATH_STATUS_AVAILABLE frames are ack-eliciting. If a packet containing a PATH_STATUS_BACKUP or PATH_STATUS_AVAILABLE frame is considered lost, the peer **SHOULD** resend the frame only if it contains the last status sent for that path -- as indicated by the sequence number.

A PATH_STATUS_BACKUP or a PATH_STATUS_AVAILABLE frame **MAY** be bundled with a PATH_NEW_CONNECTION_ID frame or a PATH_RESPONSE frame in order to indicate the preferred path usage before or during path initiation.

## 4.4. PATH_NEW_CONNECTION_ID Frame

The PATH_NEW_CONNECTION_ID frame (type=0x3e78) is an extension of the NEW_CONNECTION_ID frame specified in Section 19.15 of [QUIC-TRANSPORT]. It is used to provide its peer with alternative connection IDs for 1-RTT packets for a specific path. The peer can then use a different connection ID on the same path to break linkability when migrating on that path; see also Section 9.5 of [QUIC-TRANSPORT].

PATH_NEW_CONNECTION_ID frames are formatted as shown in Figure 10.

```
PATH_NEW_CONNECTION_ID Frame {
  Type (i) = 0x3e78,
  Path Identifier (i),
  Sequence Number (i),
  Retire Prior To (i),
  Length (8),
  Connection ID (8..160),
  Stateless Reset Token (128),
}
```

*Figure 10: PATH_NEW_CONNECTION_ID Frame Format*

Compared to the NEW_CONNECTION_ID frame specified in Section 19.15 of [QUIC-TRANSPORT], the following field is added:

Path Identifier: The path ID associated with the connection ID. This means the provided connection ID can only be used on the corresponding path.

Note that, other than for the NEW_CONNECTION_ID frame in Section 19.15 of [QUIC-TRANSPORT], the sequence number applies on a per-path context. This means different connection IDs on different paths might have the same sequence number value.

The Retire Prior To field indicates which connection IDs should be retired among those that share the path ID in the Path Identifier field. Connection IDs associated with different path IDs are not affected.

Note that the NEW_CONNECTION_ID frame can only be used to issue or retire connection IDs for the initial path with path ID 0.

The last paragraph of Section 5.1.2 of [QUIC-TRANSPORT] specifies how to verify the Retire Prior To field of an incoming NEW_CONNECTION_ID frame. The same rule applies for PATH_NEW_CONNECTION_ID frames, but it applies per path. If the multipath extension is used, the rule for the NEW_CONNECTION_ID frame is only applied for path ID 0.

## 4.5. PATH_RETIRE_CONNECTION_ID Frame

The PATH_RETIRE_CONNECTION_ID frame (type=0x3e79) is an extension of the RETIRE_CONNECTION_ID frame specified in Section 19.16 of [QUIC-TRANSPORT]. It is used to indicate that an endpoint will no longer use a connection ID for a specific path ID that was issued by its peer. To retire the connection ID used during the handshake on the initial path, path ID 0 is used. Sending a PATH_RETIRE_CONNECTION_ID frame also serves as a request to the peer to send additional connection IDs for this path (see also Section 5.1 of [QUIC-TRANSPORT]), unless the path specified by the path ID has been abandoned. New path-specific connection IDs can be delivered to a peer using the PATH_NEW_CONNECTION_ID frame (see Section 4.4).

PATH_RETIRE_CONNECTION_ID frames are formatted as shown in Figure 11.

```
PATH_RETIRE_CONNECTION_ID Frame {
  Type (i) = 0x3e79,
  Path Identifier (i),
  Sequence Number (i),
}
```

*Figure 11: PATH_RETIRE_CONNECTION_ID Frame Format*

Compared to the RETIRE_CONNECTION_ID frame specified in Section 19.16 of [QUIC-TRANSPORT], the following field is added:

Path Identifier: The path ID associated with the connection ID to retire.

Note that the RETIRE_CONNECTION_ID frame can only be used to retire connection IDs for the initial path with path ID 0.

As the PATH_NEW_CONNECTION_ID frame applies the sequence number per path, the sequence number in the PATH_RETIRE_CONNECTION_ID frame is also per path. The PATH_RETIRE_CONNECTION_ID frame retires the Connection ID with the specified path ID and sequence number.

The processing of an incoming RETIRE_CONNECTION_ID frame is described in Section 19.16 of [QUIC-TRANSPORT]. The same processing applies for PATH_RETIRE_CONNECTION_ID frames per path, while with use of the multipath extension, the processing of a RETIRE_CONNECTION_ID frame is only applied for path ID 0.

## 4.6. MAX_PATH_ID Frame

A MAX_PATH_ID frame (type=0x3e7a) informs the peer of the maximum path ID it is permitted to use.

MAX_PATH_ID frames are formatted as shown in Figure 12.

```
MAX_PATH_ID Frame {
  Type (i) = 0x3e7a,
  Maximum Path Identifier (i),
}
```

*Figure 12: MAX_PATH_ID Frame Format*

MAX_PATH_ID frames contain the following field:

Maximum Path Identifier: The maximum path ID that the sending endpoint is willing to accept. This value **MUST NOT** exceed $2^{32}$-1, which is the maximum allowed value for the path ID due to restrictions on the nonce calculation (see Section 2.4). The Maximum Path Identifier value **MUST NOT** be lower than the value advertised in the initial_max_path_id transport parameter.

Receipt of an invalid Maximum Path Identifier value **MUST** be treated as a connection error of type PROTOCOL_VIOLATION.

Loss or reordering can cause an endpoint to receive a MAX_PATH_ID frame with a smaller Maximum Path Identifier value than was previously received. MAX_PATH_ID frames that do not increase the path limit **MUST** be ignored.

MAX_PATH_ID frames are ack-eliciting and **SHOULD** be retransmitted when lost and no more recent MAX_PATH_ID frame has been sent in the meantime.

## 4.7. PATHS_BLOCKED and PATH_CIDS_BLOCKED Frames

A sender can send a PATHS_BLOCKED frame (type=0x3e7b) when it wishes to open a path but is unable to do so due to the maximum path ID limit set by its peer.

A sender can send a PATH_CIDS_BLOCKED frame (type=0x3e7c) when it wishes to open a path with a valid path ID or change the connection ID on an established path but is unable to do so because there are no unused connection IDs available for the corresponding path ID.

Note that PATHS_BLOCKED and PATH_CIDS_BLOCKED frames are informational. Sending a PATHS_BLOCKED or a PATH_CIDS_BLOCKED frame does not imply a particular action from the peer like sending a MAX_PATH_ID frame with a new Maximum Path Identifier value but informs the peer that the maximum path ID limit or the absence of unused connection IDs prevented the creation or the usage of paths. If the successful reception of a PATHS_BLOCKED/ PATH_CIDS_BLOCKED frame was acknowledged, but no action is taken by the peer, this is likely a deliberate decision by the peer and repeating the PATHS_BLOCKED/PATH_CIDS_BLOCKED frame will not change that.

PATHS_BLOCKED frames are formatted as shown in Figure 13.

```
PATHS_BLOCKED Frame {
  Type (i) = 0x3e7b,
  Maximum Path Identifier (i),
}
```

*Figure 13: PATHS_BLOCKED Frame Format*

PATHS_BLOCKED frames contain the following field:

Maximum Path Identifier: A variable-length integer indicating the maximum path ID that was allowed at the time the frame was sent. If the received value is lower than the currently allowed maximum value, this frame can be ignored.

PATH_CIDS_BLOCKED frames are formatted as shown in Figure 14.

```
PATH_CIDS_BLOCKED Frame {
  Type (i) = 0x3e7c,
  Path Identifier (i),
  Next Sequence Number (i),
}
```

*Figure 14: PATH_CIDS_BLOCKED Frame Format*

PATH_CIDS_BLOCKED frames contain the following fields:

Path Identifier: Identifier of the path for which unused connection IDs are not available.

Next Sequence Number: The next sequence number that is expected to be issued for a connection ID for this path by the peer.

Receipt of a value of Maximum Path Identifier or Path Identifier that is higher than the local maximum value **MUST** be treated as a connection error of type PROTOCOL_VIOLATION.

Receipt of a value of Next Sequence Number that is higher than the sequence number of the next expected to be issued connection ID for this path **MUST** be treated as a connection error of type PROTOCOL_VIOLATION.

PATHS_BLOCKED and PATH_CIDS_BLOCKED frames are ack-eliciting and **MAY** be retransmitted if the path is still blocked when the loss is detected.

# 5. Implementation Considerations

This section provides informational guidance for implementors.

## 5.1. Connection ID Changes, Migration, and NAT Rebindings

With the multipath extension, each path uses a separate packet number space. This is a major difference from [QUIC-TRANSPORT], which only defines three number spaces (Initial, Handshake, and Application data packets).

For any given path, connection ID rotation, NAT rebinding, or client-initiated migration as specified in [QUIC-TRANSPORT] might occur like on a single path. These events do not change the path ID and do not affect the packet number space associated with the path.

It is generally preferable to use multipath mechanisms, such as creating a new path and later abandoning the old path rather than doing migration of a single path as specified in [QUIC-TRANSPORT]. This enables a smoother handover and allows a more controlled migration handling at the server side. However, migration of a single path cannot be avoided in case of NAT rebinding or if the server requests migration to a "preferred address" during the handshake.

Section 9.3 of [QUIC-TRANSPORT] allows an endpoint to skip validation of a peer address if that address has been seen recently. However, when the multipath extension is used and an endpoint has multiple addresses that could lead to switching between different paths, it should rather maintain multiple open paths instead.

Servers observing a 4-tuple change will perform path validation (see Section 9 of [QUIC-TRANSPORT]). If path validation process succeeds, the endpoints reset the path's congestion controller and round-trip time estimator according to Section 9.4 of [QUIC-TRANSPORT].

## 5.2. Using Multiple Paths on the Same 4-tuple

It is possible to create paths that refer to the same 4-tuple. For example, endpoints might want to create paths that use different Differentiated Service [RFC2475] markings. This could be done in conjunction with scheduling algorithms that match streams to paths so that, for example, data frames for low priority streams are sent over low priority paths. Since these paths use different path IDs, they can be managed independently to suit the needs of the application. (The application would need to manage how client and server use differentiated services on a path. This is not specified in this document.)

There might be cases in which paths are created with different 4-tuples but end up using the same 4-tuples as a consequence of path migrations. Consider the following example where all paths use the same source and destination ports:

- Client starts path 1 from address 192.0.2.1 to server address 198.51.100.1.
- Client starts path 2 from address 192.0.2.2 to server address 198.51.100.1.
- Both paths are used for a while.
- Server sends packet from address 198.51.100.1 to client address 192.0.2.1 with Connection ID indicating path ID 2.
- Client receives the packet, recognizes a path migration, and updates the source address of path 2 to 192.0.2.1.

(For simplicity, this example uses IPv4 addresses but would work similarly for IPv6 addresses.)

Such unintentional use of the same 4-tuple on different paths ought to be rare. When they happen, the two paths would be redundant, and the endpoint could want to close one of them.

## 5.3. Congestion Control

When the QUIC multipath extension is used, senders manage per-path congestion status as required in Section 9.4 of [QUIC-TRANSPORT]. However, in [QUIC-TRANSPORT], only one active path is assumed and, as such, the requirement is to reset the congestion control status on path migration. With the multipath extension, multiple paths can be used simultaneously. Therefore, separate congestion control state is maintained for each path. This means a sender is not allowed to send more data on a given path than congestion control for that path indicates.

When a Multipath QUIC connection uses two or more paths, there is no guarantee that these paths are fully disjoint. When two (or more paths) share the same bottleneck, using a standard congestion control scheme could result in an unfair distribution of the bandwidth with the multipath connection getting more bandwidth than competing single paths connections. Multipath TCP uses the Linked Increases Algorithm (LIA) congestion control scheme specified in [RFC6356] to solve this problem. This scheme can immediately be adapted to Multipath QUIC. Other coupled congestion control schemes have been proposed for Multipath TCP, such as [OLIA]. Designers of congestion control algorithms specialized for Multipath QUIC are advised to follow BCP 133; see Section 7.10 of [RFC9743].

Section 5.1.2 of [QUIC-TRANSPORT] indicates that an endpoint can change the connection ID it uses to another available one at any time during the connection. As such, a change of the Connection ID without any change in the address does not indicate a path change and the endpoint can keep the same congestion control and RTT measurement state.

## 5.4. Computing Path RTT

PATH_ACK frames indicate which path the acknowledged packets were sent on, but they could be received through any open path. If successive acknowledgments are received on different paths, the measured RTT samples can fluctuate widely, which could result in poor performance depending on, for example, the congestion control algorithm.

Congestion control state as defined in [QUIC-RECOVERY] is kept per path ID. However, depending on which path acknowledgments are sent, the actual RTT of a path cannot be calculated or might not be the right value to be used.

Instead of using the real RTT of a path, it is recommended to consider the sum of two one-way delays: the delay on the packet sending path and the delay on the return path chosen for the acknowledgments. When different paths have different characteristics, the delays can vary widely. For example, consider a 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 each direction. The sum of the two one-way delays will depend on the combination of paths used for the packet transmission and the acknowledgment 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*

The computed values reflect both the state of the network path and the scheduling decisions of the acknowledgment sender. If we assume that the PATH_ACK will be sent over the terrestrial link, because this decision provides the best response time, the computed RTT value for the satellite path will be about 350ms. This is lower than the 600ms that would be measured if the PATH_ACK came over the satellite channel, but it is still the right value for computing, for example, the PTO timeout: If a PATH_ACK is not received after more than 350ms, either the packet or its PATH_ACK were probably lost.

The simplest implementation is to use the delays measured when receiving new packet acknowledgments to compute smoothed_rtt and rttvar per Section 5.3 of [QUIC-RECOVERY] regardless of the path through which PATH_ACK frames are received. This approach will provide good results as long as acknowledgments are sent consistently over one path. If at any time the acknowledgment sender revisits its sending preferences, this can also change the paths that are used to send acknowledgments. However, this is not very different from route changes on a single path. The RTT, RTT variance, and PTO estimates will rapidly converge to reflect the new conditions. There is one exception: the minimum RTT, which is also a known challenge when route changes occurs on a single path. An acknowledgment receiver can, however, remember the path over which the PATH_ACK that produced the minimum RTT was received and restart the minimum RTT computation if that acknowledgment path changes or is abandoned. If acknowledgments are not sent consistently over one path, the acknowledgment receiver can monitor over which path acknowledgments are received and only use samples for acknowledgments received on the same path on which the data was sent, if any.

Further, congestion control functions that rely on delay estimates need to consider cases where acknowledgments are sent over multiple paths with different delays explicitly.

## 5.5. Packet Scheduling

The transmission of packets containing data is limited by the arrival of data from the application and by congestion control. Generally, QUIC packets that increase the number of bytes in flight can only be sent when the congestion window for the selected path allows it.

Most frames, including control frames (PATH_CHALLENGE and PATH_RESPONSE being the notable exceptions), can be sent and received on any open path. As such, a packet scheduler is needed to decide which path to use for sending the next packet, among those paths with an open congestion window. If multiple paths are used to send data frames belonging to the same stream, data delivery will experience the maximum delay of all used paths due to in-order delivery. The scheduling is a local decision based on the preferences of the application and the implementation.

This implies that an endpoint might send and receive PATH_ACK frames on a path different from the one that carried the acknowledged packets. As noted in Section 5.4, RTT estimates computed using the standard algorithm reflect both the characteristics of the path and the scheduling algorithm of PATH_ACK frames. The estimates will converge faster if the scheduling strategy of PATH_ACK frames is stable. Implementations can choose different strategies such as, for instance, sending PATH_ACK frames either simply on the path where the acknowledged packets were received or alternatively the shortest path, which results in shorter control loops and potentially better performance.

Since packets that only carry PATH_ACK frames are not congestion controlled (see Section 7 of [QUIC-RECOVERY]), senders should carefully consider the load induced by these packets, especially if the capacity is unknown on that path, e.g., when that path is not used for sending data frames.

## 5.6. Retransmissions

Simultaneous use of multiple paths enables different retransmission strategies to cope with losses such as a) retransmitting lost frames over the same path, b) retransmitting lost frames on a different or dedicated path, and c) duplicate lost frames on several paths (not recommended for general purpose use due to the network overhead). While this document does not preclude a specific strategy, more detailed specification is out of scope.

As noted in Section 2.2 of [QUIC-TRANSPORT], STREAM frame boundaries are not expected to be preserved when data is retransmitted. Especially when STREAM frames have to be retransmitted over a different path with a smaller MTU limit, smaller STREAM frames might need to be sent instead.

## 5.7. PTO Expiration

An implementation should follow the mechanism specified in [QUIC-RECOVERY] for detecting packet loss on each individual path. A special case happens when the PTO timer expires. According to [QUIC-RECOVERY], no packet will be declared lost until either the packet sender

receives a new acknowledgment for this path or the path itself is finally declared broken. This cautious process minimizes the risk of spurious retransmissions, but it might cause significant delivery delay for the frames contained in these "lost packets".

Endpoints could take advantage of the multipath extension and retransmit the content of the delayed packets on other available paths if the congestion control window on these paths allows.

### 5.8. Paths Having Different PMTU Sizes

An implementation should take care to handle different Path MTU (PMTU) sizes across multiple paths. As specified in Section 14.3 of [QUIC-TRANSPORT], the Datagram Packetization Layer PMTU Discovery (DPLPMTUD) Maximum Packet Size (MPS) is maintained for each combination of local and remote IP addresses. Note that with the multipath extension, multiple paths could use the same 4-tuple but might have different MPS due to other factors (see Section 5.2).

One simple option, if the PMTUs are similar, is to apply the minimum PMTU of all paths to each path, which could also help to simplify retransmission processing.

### 5.9. Idle Timeout and Keep-Alives

[QUIC-TRANSPORT] defines an idle timeout for closing the connection, which applies in case of multipath usage if no packet is received on any path for the duration of the idle timeout.

This document does not specify per-path idle timeouts. An endpoint can decide to close a path at any time, whether the path is in active use or not. For example, an endpoint might wait to send the initial PATH_ABANDON frame until it sends another frame anyway. Note that the receiver of an initial PATH_ABANDON frame is, however, required to immediately reply (see Section 3.4).

If a path is not actively used for a while, it might not be usable anymore, e.g., due to middlebox timeouts. To avoid such path breakage, endpoints can send ack-eliciting packets, such as packets containing PING frames (Section 19.2 of [QUIC-TRANSPORT]) on that path to keep it alive. However, this specification does not recommend sending keep-alives, as it can create unnecessary overhead, especially if there are other actively used paths.

Section 5.3 of [QUIC-TRANSPORT] defines an optional keep-alive process. This process can be applied to each path separately depending on application needs. Some applications could decide to not keep any not-actively used path alive or keep only one additional path or multiple paths (e.g., for more redundancy) alive. As discussed in Section 10.1.2 of [QUIC-TRANSPORT], the keep-alive interval needs to incorporate timeouts in middleboxes on the path.

If a path was not actively used for a while and no keep-alives have been sent, an endpoint can probe it before switching to active use if there are still other paths that are currently usable.

## 6. IANA Considerations

This document defines a new transport parameter to enable simultaneous use of multiple paths within one QUIC connection. Further, it specifies new frame types for path management and new error codes when a path is abandoned.

IANA has allocated short values with permanent as the status, IETF as the change controller, and the QUIC WG as the contact to the respective registries under <https://www.iana.org/assignments/quic>.

The following parameter defined in Table 2 has been added to the "QUIC Transport Parameters" registry.

| Value | Parameter Name | Specification |
|---|---|---|
| 0x3e | initial_max_path_id | Section 2.1 |

*Table 2: Addition to QUIC Transport Parameters Entries*

The following frame types defined in Table 3 have been added to the "QUIC Frame Types" registry.

| Value | Frame Type Name | Specification |
|---|---|---|
| 0x3e-0x3f | PATH_ACK | Section 4.1 |
| 0x3e75 | PATH_ABANDON | Section 4.2 |
| 0x3e76 | PATH_STATUS_BACKUP | Section 4.3 |
| 0x3e77 | PATH_STATUS_AVAILABLE | Section 4.3 |
| 0x3e78 | PATH_NEW_CONNECTION_ID | Section 4.4 |
| 0x3e79 | PATH_RETIRE_CONNECTION_ID | Section 4.5 |
| 0x3e7a | MAX_PATH_ID | Section 4.6 |
| 0x3e7b | PATHS_BLOCKED | Section 4.7 |
| 0x3e7c | PATH_CIDS_BLOCKED | Section 4.7 |

*Table 3: Addition to QUIC Frame Types Entries*

The following transport error codes defined in Table 4 have been added to the "QUIC Transport Error Codes" registry.

| Value | Code | Description | Specification |
|---|---|---|---|
| 0x3e | APPLICATION_ABANDON_PATH | Path abandoned at the application's request | Section 4.2.1 |
| 0x3e75 | PATH_RESOURCE_LIMIT_REACHED | Path abandoned due to resource limitations in the transport | Section 4.2.1 |
| 0x3e76 | PATH_UNSTABLE_OR_POOR | Path abandoned due to unstable interfaces or poor performance | Section 4.2.1 |
| 0x3e77 | NO_CID_AVAILABLE_FOR_PATH | Path abandoned due to no available connection IDs for the path | Section 4.2.1 |

*Table 4: Addition to QUIC Transport Error Codes Entries*

# 7. Security Considerations

The multipath extension retains all security properties of [QUIC-TRANSPORT] and [QUIC-TLS] but requires some additional consideration regarding:

- potential additional resource usage for per-path connection IDs and multiple concurrent path contexts,
- a potentially increased amplification risk for denial-of-service attacks if multiple paths are used simultaneously, and
- changes to the nonce calculation due to the use of multiple packet number spaces.

## 7.1. Memory Allocation for Per-Path Resources

The maximum path ID limit in initial_max_path_id or the MAX_PATH_ID frame limits the number of paths an endpoint is willing to maintain and thereby also limits the associated path resources. Furthermore, as connection IDs have to be issued by both endpoints for the same path ID before an endpoint can open a path, each endpoint could also control the per-path resource usage by only issuing connection IDs for a limited number of paths. However, using the maximum path ID limit in initial_max_path_id or the MAX_PATH_ID frame is preferred.

To avoid unnecessary resource usage that could be exploited in a resource exhaustion attack, endpoints **SHOULD** allocate additional path resources, such as, e.g., for packet number handling, only after path validation has successfully completed.

### 7.2. Denial of Service with Multiple Paths

Path validation as specified in Section 8.2 of [QUIC-TRANSPORT] for migration is used unchanged for path initiation in this extension. Further, the multipath extension allows for the creation of multiple paths, which means that in addition to the security considerations on source address spoofing outlined in Section 21.5.4 of [QUIC-TRANSPORT], there is a risk of amplified denial-of-service attacks through simultaneous opening or migration of multiple paths. For example, an attacker could set or spoof the 4-tuples used in multiple paths so that packets sent by the server would travel through common network paths in an attempt to overwhelm a target.

[QUIC-TRANSPORT] only allows the use of one path and the number of concurrent path validation attempts is limited by the number of issued connection IDs. This extension allows for multiple open paths that could in theory be migrated all at the same time, still limited by the number of issued connection IDs for each path. Further, multiple paths could be initialized simultaneously, limited by the maximum number of allowed paths.

Therefore, endpoints are advised to keep the maximum number of paths small and might consider additional measures to limit the number of concurrent path validation processes, e.g., by pacing them out or limiting the number of path initiation attempts over a certain time period.

However, the use of multipath does not change the anti-amplification limits as specified in Section 8 of [QUIC-TRANSPORT]. The attacker would need to send a separate challenge on each path used in the attack, and the amplification limits would apply to the amount of data sent by the server on that path. The attacker could get the same effect by opening many QUIC connections and conducting the attack on each of them without negotiating the multipath option.

### 7.3. Cryptographic Handshake and AEAD Nonce

The multipath extension as specified in this document is only enabled after a successful handshake when both endpoints indicate support for this extension. All new frames defined in this extension are only used in 1-RTT packets.

As the handshake is not changed by this extension, the transport security mechanisms as specified in [QUIC-TLS], such as encryption key exchange and peer authentication, remain unchanged. As such, the security considerations in [QUIC-TLS] apply unaltered.

The limits as discussed on Appendix B of [QUIC-TLS] apply to the total number of packets sent on all paths, not each path separately.

This specification changes the AEAD nonce calculation by including the path ID as part of the calculation (see Section 2.4). To ensure unique nonces, path IDs are limited to 32 bits and cannot be reused for another path of the same connection.

## 8. References

### 8.1. Normative References

**[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/info/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/info/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/info/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>.

### 8.2. Informative References

**[OLIA]** Khalili, R., Gast, N., Popovic, M., Upadhyay, U., and J. Le Boudec, "MPTCP is not pareto-optimal: performance issues and a possible solution", CoNEXT '12: Proceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 1-12, DOI 10.1145/2413176.2413178, December 2012, <https://doi.org/10.1145/2413176.2413178>.

**[RFC2475]** Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

**[RFC6356]** Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <https://www.rfc-editor.org/info/rfc6356>.

**[RFC9312]** Kühlewind, M. and B. Trammell, "Manageability of the QUIC Transport Protocol", RFC 9312, DOI 10.17487/RFC9312, September 2022, <https://www.rfc-editor.org/info/rfc9312>.

**[RFC9743]** Duke, M., Ed. and G. Fairhurst, Ed., "Specifying New Congestion Control Algorithms", BCP 133, RFC 9743, DOI 10.17487/RFC9743, March 2025, <https://www.rfc-editor.org/info/rfc9743>.

## Acknowledgments

This document is a collaboration of authors that combines work from three proposals. Further authors of one of the original proposals are Qing An and Zhenyu Li.

Thanks to Marten Seemann, Kazuho Oku, Martin Thomson, Magnus Westerlund, Mike Bishop, Lucas Pardue, Michael Eriksson, Yu Zhu, Gorry Fairhurst, Tilmann Zäschke, and Tommy Pauly for their thorough reviews and valuable contributions.

Thanks for IETF Last Call reviews to Carsten Bormann (artart), Antoine Fressancourt (intdir), Adrian Farrel (opsdir), and Meral Shirazipour (genart).

## Authors' Addresses

**Yanmei Liu (EDITOR)**
Alibaba Inc.
Email: miaoji.lym@alibaba-inc.com

Additional contact information:

**刘彦梅 (EDITOR)**
Alibaba Inc.

**Yunfei Ma**
Harvey AI
Email: yfmascgy@gmail.com

Additional contact information:

**马云飞**
Harvey AI

**Quentin De Coninck (EDITOR)**
University of Mons (UMONS)
Email: quentin.deconinck@umons.ac.be

**Olivier Bonaventure**
UCLouvain and WELRI
Email: olivier.bonaventure@uclouvain.be

**Christian Huitema**
Private Octopus Inc.
Email: huitema@huitema.net

**Mirja Kühlewind (EDITOR)**
Ericsson
Email: mirja.kuehlewind@ericsson.com