Multipath Extensions for QUIC (MP-QUIC)
draft-deconinck-quic-multipath-03

Abstract

This document specifies extensions to the QUIC protocol to enable, other than connection migration, simultaneous usage of multiple paths for a single connection. Those extensions remain compliant with the current single-path QUIC design. They allow devices to benefit from multiple network paths while preserving the privacy features of QUIC.

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 March 1, 2020.

Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

Endhosts have evolved. Today’s endhosts are equipped with several network interfaces and users expect to be able to seamlessly switch from one to another or use them simultaneously to aggregate or grab bandwidth whenever needed. During the last years, several multipath extensions to transport protocols have been proposed [RFC6824], [MPRT], [SCTPCMT]. Multipath TCP [RFC6824] is the most mature one. It is already deployed on popular smartphones, but also for other use cases [RFC8041].

With regular TCP and UDP, all the packets that belong to a given flow share the same 5-tuple that acts as an identifier for this flow. Such characterization prevents these flows from using multiple paths. QUIC [I-D.ietf-quic-transport] does not use the 5-tuple as an implicit connection identifier. A QUIC flow is identified by two Connection IDs (Source and Destination). This enables QUIC flows to cope with events affecting the 5-tuple, such as NAT rebinding or IP address changes. The QUIC connection migration feature, described in more details in [I-D.ietf-quic-transport], is key to migrate a flow from one 5-tuple to another one. This migration feature offers the opportunity for QUIC to sustain a connection over multiple paths, but still there is a void to specify simultaneous usage of available paths for a single connection. Use cases such as bandwidth aggregation or seamless network handovers would be applicable to QUIC, as they are now with Multipath TCP. An early performance evaluation of such use cases and a comparison between Multipath QUIC and Multipath TCP may be found in [MPQUIC].

In this document, we leverage many of the lessons learned from the design of Multipath TCP and the comments received on the first versions of this draft to propose extensions to the current QUIC design to enable it to simultaneously use several paths. This document focuses mainly on paths that are locally visible to an endpoint. This document is organized as follows. It first provides in Section 3 an overview of the operation of Multipath QUIC. It then states changes required in the current QUIC design [I-D.ietf-quic-transport] and specifies in Section 5 the usage of multiple paths. Finally, it discusses some security considerations.
2. Conventions and Definitions

The words "MUST", "MUST NOT", "SHOULD", and "MAY" are used in this document. It’s not shouting; when they are capitalized, they have the special meaning defined in [RFC2119].

We assume that the reader is familiar with the terminology used in [I-D.ietf-quic-transport]. In addition, we define the following terms:

- **Path**: A logical association seen by a QUIC host to reach its peer, on which packets can be sent. It is typically characterized by (Source IP Address, Source Port Number, Destination IP Address, Destination Port Number). A path is unidirectional, similar to the Connection IDs. A path is internally identified by using a Path ID and uses a potentially changing Connection ID identifying the parent connection in packets exchanged on that path.

- **Initial Paths**: The paths used by peers for the establishment of the QUIC connection. The cryptographic handshake is done on these paths. These are identified by Path ID 0.

2.1. Notational Conventions

Packet and frame diagrams use the format described in Section 2.1 of [I-D.ietf-quic-transport].

3. Overview

The current design of QUIC [I-D.ietf-quic-transport] provides reliable transport with multiplexing and security. A wide range of devices on today’s Internet are multihomed. Examples include smartphones equipped with both WLAN and cellular interfaces, but also regular dual-stack hosts that use both IPv4 and IPv6. Experience with Multipath TCP has shown that the ability to combine different paths during the lifetime of a connection provides various benefits including bandwidth aggregation or seamless handovers [RFC8041],[IETFJ].

The current design of QUIC does not enable multihomed devices to efficiently use different paths simultaneously. We first explain why a multipath extension would be beneficial to QUIC and then describe it at a high level.
3.1. What is a Path?

Before going into details, let us first define what is called a "path". Since the early days of computer networks, a path was
denoted by a 4-tuple (Source IP Address, Source Port Number,
Destination IP Address, Destination Port Number). It is namely a UDP
road from the local host to the remote one. Considering a smartphone
interacting with a single-homed server, the smartphone might want to
use one path over the WLAN network and another over the cellular one.
Those paths are not necessarily disjoint. For example, when
interacting with a dual-stack server, a smartphone may create two
paths over the Wi-Fi network, one over IPv4 and the other over IPv6.
However, networking experiences showed that packets following a
direction do not always share the same road as the packets in the
opposite direction. Therefore, QUIC paths are host-specific, i.e.,
the path from A to B is different from the one from B to A. This
potentially enables the use of unidirectional networks such as
satellites, unapplicable when using TCP.

3.2. Beyond Connection Migration

Unlike TCP [RFC0793], QUIC is not bound to a particular 4-tuple
during the lifetime of a connection. A QUIC connection is identified
by a Connection ID, placed in the public header of each QUIC packet.
This enables hosts to continue the connection even if the 4-tuple
changes due to, e.g., NAT rebinding. This ability to shift a
connection from one 4-tuple to another is called Connection
Migration. One of its use cases is fail-over when the IP address in
use fails but another one is available. A device losing the WLAN
connectivity can then continue the connection over its cellular
interface, for instance.

A QUIC peer can thus start on a given path, denoted as the initial
path, and end on another one. However, the current QUIC design
[I-D.ietf-quic-transport] assumes that only one symmetric path is in
use for a given connection. The specification does not support means
distinct path migration from simultaneous usage of available
asymmetric paths for a given connection.

This document fills that void. Concretely, instead of switching the
4-tuple for the whole connection, this draft first proposes
mechanisms to communicate endhost addresses to the peer. It then
leverages the address validation process with the PATH_CHALLENGE and
PATH_RESPONSE frames proposed in [I-D.ietf-quic-transport] to verify
if additional addresses advertised by the communicating host are
available and actually belong to it. In this case, those addresses
can be used to create new paths to spread packets over several
networks following a traffic distribution policy that is out of scope of this document.

When multiple paths are available, different delays may be experienced as a function of the initial path selected for the establishment of the QUIC connection. The first version of the specification does not discuss considerations related to the selection of the initial path to place the connection.

The example of Figure 1 shows a possible data exchange between a dual-homed client performing a request fitting in two packets and a single-homed server. Notice that the Path ID used by an host over a specific network is not necessarily the same as the one used by the remote peer. In the presented example, phone sends packets over WLAN on path 1 and over LTE on path 2, while the packets sent by the server over WLAN are on path 2 and over LTE are on path 1.

![Figure 1: Dataflow with Multipath QUIC](image)

The remaining of this section focuses on providing a high-level overview of the multipath operations in QUIC.
3.3. Establishment of a Multipath QUIC Connection

A Multipath QUIC connection starts like a regular QUIC connection. A cryptographic handshake takes place with CRYPTO frames and follows the classical process [I-D.ietf-quic-transport] [I-D.ietf-quic-tls]. It is during that process that the multipath capability is negotiated between hosts. This is performed using the max_sending_paths transport parameter, where both hosts advertise their support for the multipath extension. Any value different from 0 indicates that the host wants to support multipath over the connection. If one of the hosts does not advertise the max_sending_paths transport parameter, the negotiated value is 0, meaning that the QUIC connection will not use the multipath extensions presented in this document.

The handshake is performed on a given path. This path is called the Initial path and is identified by Path ID 0.

3.4. Architecture of Multipath QUIC

Once established, a Multipath QUIC connection is composed of two or more asymmetric paths. Each asymmetric path is associated with a different four-tuple and identified by a Path ID, as shown in Figure 2.

```
+-----------------------------------------------+    +-----------------------------------------------+
|           Connection (MSCID, MDCID)           |    |           Connection (MSCID, MDCID)           |
+-----------------------------------------------+    +-----------------------------------------------+
| Sending    | Sending    | ... | Sending    | Sending    |
| Path 0     | Path 1     | ... | Path N - 1 |
| Tuple      | Tuple'     | ... | Tuple"    |
| PDCID      | PDCID'     | ... | PDCID"    |
| PN         | PN'        | ... | PN"       |
+-----------------------------------------------+    +-----------------------------------------------+
| Receive    | Receive    | ... | Receive    |
| Path 0     | Path 1     | ... | Path M - 1 |
| Tuple      | Tuple'     | ... | Tuple"    |
| PSCID      | PSCID'     | ... | PSCID"    |
| PN         | PN'        | ... | PN"       |
+-----------------------------------------------+    +-----------------------------------------------+
```

Figure 2: Architectural view of Multipath QUIC

As described before, a Multipath QUIC connection starts on the Initial Path, identified by Path ID 0. For Multipath QUIC, this document proposes two levels of asymmetric Connection IDs. The first ones are the Main (or Primary) Connection IDs (MCIDs). Both the Main
Source Connection ID (MSCID) and the Main Destination Connection ID (MDCID) uniquely identify the connection, as with the current QUIC design. The second ones are the Path Connection IDs (PCIDs), written in the Connection ID field of the public header. The PCIDs act as the path identifiers for packets. Depending on the direction of the path (either sending path or receiving one), the host keeps either the Path Source Connection ID (PSCID, for the receive paths) or the Path Destination Connection ID (PDCID, for the sending paths).

Notice that the PDCID of a sending path of an host is the same as the PSCID of the corresponding receive path of the remote. Preventing the linkability of different paths is an important requirement for the multipath extension [I-D.huitema-quic-mpath-req]. Using PCIDs as implicit path identifier makes this linkability harder than having explicit signaling as in the early version of this draft and does not require public header change to keep invariants [I-D.ietf-quic-invariants]. The MCIDs of a connection will be the PCIDs of the Initial Path. In the example of Figure 1, if the connection started using WLAN, then the Destination Connection ID A is both the PDCID of the WLAN sending path and the MDCID of the connection.

In addition to the PCIDs, some additional information is kept for each asymmetric path. The Path ID identifies the asymmetric path at the frame level and ensures uniqueness of the nonce (see Section 8.1 for details). A congestion window is maintained for each sending path. Hosts can also collect network measurements on a per-path basis, such as one-way delay measurements and lost packets.

3.5. Path Establishment

The max_sending_paths transport parameter exchanged during the cryptographic handshake determines if multiple paths can be used. Then, hosts provide to their peer the Path Connection ID to use on asymmetric paths. Unlike Multipath TCP [RFC6824], both hosts dynamically control how many sending paths can currently be in use by the peer, i.e., the number of different Path IDs proposed to the peer. Notice that the peers might advertise a different number of sending Path IDs to their peer, setting different limits to the sending and receive paths of each host. The max_sending_paths transport parameter indicates the maximum number of sending paths the host would support to receive from the peer.

Hosts propose new receive paths with an extended version of the NEW_CONNECTION_ID frame (see Section 6.1). This frame provides to the peer the PDCID for a given sending path. Upon reception of the frame, the peer can start using the proposed sending path Path ID with the provided PDCID. Therefore, once an host sent such NEW_CONNECTION_ID frame, it MUST be ready to receive packets from
that Path ID with the proposed PCID. As frames are encrypted, adding new paths does not leak cleartext identifiers
[I-D.huitema-quic-mpath-req].

A server might provide several Path Connection IDs for the same asymmetric Path IDs with multiple NEW_CONNECTION_ID frames. This can be useful to cope with migration cases, as described in Section 3.9. Multipath QUIC is asymmetrical. Any host can start using new sending paths once their corresponding PDCIDs have been provided by the remote peer.

Hosts are not able to create new paths as long as the peer does not send NEW_CONNECTION_ID frames. To limit the latency of the path handshake, hosts should send those frames as soon as possible, i.e., just after the 0-RTT handshake packet.

Sending useful data on a fresh new sending path might lead to poor performance as the network path used by the QUIC path is not usable yet. A typical case is when a server wants to initiate a new path to a client behind a NAT. The client would possibly never receive this packet, leading to connectivity issues on that path. To avoid such issues, a remote address MUST have been validated as described in [I-D.ietf-quic-transport] before sending packets on a sending path using it. A host MUST be prepared to receive packets on paths it advertised.

Because attaching to new networks may be volatile and an endpoint does not have full visibility on multiple paths that may be available (e.g., hosts connected to a CPE), an MP-QUIC capable endhost SHOULD advertise a max_sending_paths value of at least 4 and SHOULD propose at least 4 receive paths to its peer.

3.6. Exchanging Data over Multiple Paths

A QUIC packet acts as a container for one of more frames. Multipath QUIC uses the same STREAM frames as QUIC to carry data. A byte offset is associated to the data payload. One of the key design decision of (Multipath) QUIC is that frames are independent of the packets carrying them. This implies that a frame transmitted over one path could be retransmitted later on another path without any change.

The path on which data is sent is a packet-level information. This means a frame can be sent regardless of the path of the packet carrying it. Furthermore, because the data offset is a frame-level information, there is no need to define additional sequence numbers to cope with reordering across paths, unlike Multipath TCP [RFC6824] that uses a Data Sequence Number at the MPTCP level. Other flow
control considerations like the stream receive window advertised by the MAX_STREAM_DATA frame remain unchanged when there are multiple paths.

However, Multipath QUIC might face reordering at packet-level when using paths having different latencies. The presence of different Path Connection IDs ensures that the packets sent over a given path will contain monotonically increasing packet numbers. To ensure more flexibility and potentially to reduce the ACK block section of the ACK frame when aggregating bandwidth of paths exhibiting different network characteristics, each path keeps its own monotonically increasing Packet Number space. This potentially allows sending up to $2 \times 2^{62} \times 2^{62}$ packets on a QUIC connection since each path has its own packet number space.

The ACK frame is also modified to allow per-path packet acknowledgments. This remains compliant with the independence between packets and frames while providing more flexibility to hosts to decide on which sending path they want to send receive path acknowledgments. Looking again at Figure 1, packets that were sent over a given sending path (e.g., the "Response 2" packet on server sending path 1 with DCID C) can be acknowledged on another sending path (here, client sending path 1 with DCID B) that does not correspond to the same underlying network to limit the latency due to ACK transmissions on high-latency paths and to enable the usage of unidirectional networks. Such scheduling decision would not have been possible in Multipath TCP [RFC6824] which must acknowledge data on the (symmetric) path it was received on.

3.7. Exchanging Addresses

When a multi-homed mobile device connects to a dual-stacked server using its IPv4 address, it is aware of its local addresses (e.g., the Wi-Fi and the cellular ones) and the IPv4 remote address used to establish the QUIC connection. If the client wants to create new paths over IPv6, it needs to learn the other addresses of the remote peer.

This is possible with the ADD_ADDRESS frames that are sent by a Multipath QUIC host to advertise its current addresses. Each advertised address is identified by an Address ID. The addresses attached to a host can vary during the lifetime of a Multipath QUIC connection. A new ADD_ADDRESS frame is transmitted when a host has a new address. This ADD_ADDRESS frame is protected as other QUIC control frames, which implies that it cannot be spoofed by attackers. The communicated address is first validated by the receiving host before it starts using it. This ensures that the address actually belongs to the peer and that the peer can send and receive packets on
that address. It also prevents hosts from launching amplification attacks to a victim address.

If the client is behind a NAT, it could announce a private address in an ADD_ADDRESS frame. In such situations, the server would not be able to validate the communicated address. The client might still use its NATed addresses to start a new sending path. To enable the server to make the link between the private and the public addresses, Multipath QUIC provides the PATHS frame that lists current active Path IDs. Notice that an host might also discover the public addresses of its peer by observing its remote IP addresses associated to the connection.

 Likewise, the client may be located behind a NAT64. As such it may announce an IPv6 address in an ADD_ADDRESS frame, that will be received over IPv4 by an IPv4-only server. The server should not discard that address, even if it is not IPv6-capable.

 An IPv6-only client may also receive from the server an ADD_ADDRESS frame which may contain an IPv4 address. The client should rely on means, such as [RFC7050] or [RFC7225], to learn the IPv6 prefix to build an IPv4-converted IPv6 address.

 A receive path is active as soon as the host has sent the NEW_CONNECTION_ID frames proposing the corresponding Path Connection IDs to its peer. A sending path is active when its has received its Path Connection IDs and its is bound to a validated 4-tuple. The PATHS frame indicates the local and remote Address IDs that the path uses. With this information, the server can then validate the public address and associate the advertised with the perceived addresses.

 Hosts that are connected behind an address sharing mechanism may collect the external IP address and port numbers assigned to the hosts and then use there addresses in the ADD_ADDRESS. Means to gather such information include, but not limited to, UPnP IGD, PCP, or STUN.

3.8. Coping with Address Removals

 During the lifetime of a QUIC connection, a host might lose some of its addresses. A concrete example is a smartphone going out of reachability of a Wi-Fi network or shutting off one of its network interfaces. Such address removals are advertised by using REMOVE_ADDRESS frames. The REMOVE_ADDRESS frame contains the Address ID of the lost address previously communicated through ADD_ADDRESS.
3.9. Path Migration

At a given time, a Multipath QUIC endpoint gathers a set of sending and receive paths, each associated to a 4-tuple. To address privacy issues due to the linkability of addresses with Connection IDs, hosts should avoid changing the 4-tuple used by a sending path. There remain situations where this change is unavoidable. These can be categorized into two groups: host-aware changes (e.g., network handover from Wi-Fi to cellular) and host-unaware changes (e.g., NAT rebinding).

For the host-aware case, let us consider the case of a Multipath QUIC connection where the client is a smartphone with both Wi-Fi and cellular. It advertised both addresses and the server currently enables only one client-sending path, the initial one. The Initial Path uses the Wi-Fi address. Then, for some reason, the Wi-Fi address becomes unusable. To preserve connectivity, the client might then decide to use the cellular address for the Initial Path. It thus sends a REMOVE_ADDRESS announcing the loss of the Wi-Fi address and a PATHS frame to inform that the Initial Path is now using the cellular address. If the cellular address validation succeeds (which could have been done as soon as the cellular address was advertised), the server can continue exchanging data through the cellular address.

However, both server and client might want to change their path used on the cellular address for privacy concerns. If the server provides an additional path (e.g., Path ID 42) through NEW_CONNECTION_ID frame at the beginning of the connection, the client can perform the path change directly and avoid using the Initial Path Connection ID on the cellular network. This can be done using the PATH_UPDATE frame. It can indicate that the host stopped to use the Initial Path to use Path ID 42 instead. This frame is placed in the first packet sent to the new sending path with its corresponding PCID. The client can then send the REMOVE_ADDRESS and PATHS frames on this new path. Compared to the previous case, it is harder to link the paths with the IP addresses to observe that they belong to the same Multipath QUIC connection.

For the host-unaware case, the situation is similar. In case of NAT rebinding, the server will observe a change in the 2-tuple (source IP, source port) of the receive path of the packet. The server first validates that the 2-tuple actually belongs to the client [I-D.ietf-quic-transport]. If it is the case, the server can send a PATH_UPDATE frame on a previously communicated but unused Path ID. The client might have sent some packets with a given PCID on a different 4-tuple, but the server did not use the given PCID on that 4-tuple. Because some on-path devices may rewrite the source IP address to forward packets via the available network attachments.
(e.g., an host located behind a multi-homed CPE), the server may inadvertently conclude that a path is not anymore valid leading thus to frequently sending PATH_UPDATE frames as a function of the traffic distribution scheme enforced by the on-path device. To prevent such behavior, the server SHOULD wait for at least X seconds to ensure this is about a connection migration and not a side effect of an on-path multi-interfaced device.

3.10. Sharing Path Policies

Some access networks are subject to a volume quota. To prevent a peer from aggressively using a given path while available resources can be freely grabbed using existing paths, it is desirable to support a signal to indicate to a remote peer how it must place data into available paths. An approach may consist in indicating in an ADD_ADDRESS the type of the interface (e.g., cellular, WLAN, fixed) through the interface-type field. The remote peer may rely on interface-type to select the path to be used for sending data. For example, fixed interfaces will be preferred over WLAN and cellular interfaces, and WLAN interface will be preferred over cellular interface.

This information might also be used to avoid draining battery for some devices.

3.11. Congestion Control

The QUIC congestion control scheme is defined in [I-D.ietf-quic-recovery]. This congestion control scheme is not suitable when several sending paths are active. Using the congestion control scheme defined in [I-D.ietf-quic-recovery] with Multipath QUIC would result in unfairness. Each sending path of a Multipath QUIC connection MUST have its own congestion window. The windows of the different paths MUST be coupled together. Multipath TCP uses the LIA congestion control scheme specified in [RFC6356]. This scheme can immediately be adapted to Multipath QUIC. Other coupled congestion control schemes have been proposed for Multipath TCP such as [OLIA].

4. Mapping Path ID to Connection IDs

As described in the overview section, hosts need to identify on which sending path packets are sent. The Path ID must then be inferred from the public header. This is done by using Path Connection IDs in addition to Main Connection IDs.

The Master Connection IDs are determined during the cryptographic handshake and actually correspond to both Connection IDs in the
current QUIC design [I-D.ietf-quic-transport]. The Path Connection IDs of the Initial Path (with Path ID 0) are equal to the Main Connection IDs. The Path Connection IDs of the other paths are determined when the NEW_CONNECTION_ID frames are exchanged.

The server MUST ensure that all advertised Path Connection IDs are available for the whole connection lifetime. Once it sends a NEW_CONNECTION_ID frame containing a PCID, both hosts can start receiving packets with the advertised Connection ID as belonging to the corresponding receive path. Hosts MUST wait until the reception of a NEW_CONNECTION_ID frame before sending packets on the corresponding sending path.

Hosts MUST ensure that they can receive packets coming from their peer using the PCID they proposed in the NEW_CONNECTION_ID frame they sent and associate it with the corresponding receive Path ID. Upon reception of the NEW_CONNECTION_ID frame, hosts MUST acknowledge it and MUST store the advertised Destination Path Connection ID and the Path ID of the proposed path.

5. Using Multiple Paths

This section describes in details the multipath QUIC operations.

5.1. Multipath Negotiation

The Multipath Negotiation takes place during the cryptographic handshake with the max_sending_paths transport parameter. A QUIC connection is initially single-path in QUIC, and all packets prior to handshake completion MUST be exchanged over the Initial Path. During this process, hosts advertise their support for multipath operations. Any value different from 0 indicates that the host supports multipath operations over the connection, i.e., the extensions defined in this document (not to be mixed with the availability of local multiple paths). If a host does not send the max_sending_paths transport parameter during the cryptographic handshake, the remote MUST assume a value of 0, leading to a single-path connection over the Initial Path. If both hosts advertise their support of the multipath extensions, NEW_CONNECTION_ID frames can later propose Path IDs that can then be used for the connection.

5.1.1. Transport Parameter Definition

An endhost MAY use the following transport parameter:

max_sending_paths (0x0020): Indicate the support of the multipath extension presented in this document, encoded as an unsigned 8-bit integer. Any value different from 0 indicates support of the
extension. The actual value indicates the number of sending paths the host advertising the value would like to receive from its peer. If absent, the value for this parameter is 0, meaning that a host omitting this transport parameter does not agree to use the multipath extension over the connection.

5.2. Coping with Additional Remote Addresses

The usage of multiple networks paths is often done using multiple IP addresses. For instance, a smartphone willing to use both Wi-Fi and LTE will use the corresponding addresses assigned by these networks. It can then safely send packets to a previously-used IP address of a server. The server can receive packets sourced with different IP addresses, but it MUST first validate the new remote IP addresses before starting sending packets to those addresses.

Similarly, additional addresses could be communicated using ADD_ADDRESS frames. Such addresses MUST be validated before starting to send packets to them. This requirement is explained in Section 8.2.

The validation of an address could be performed with PATH_CHALLENGE and PATH_RESPONSE frames as described in [I-D.ietf-quic-transport]. A validated address MAY be cached for a given host for a limited amount of time.

5.3. Receive Path State

When proposing asymmetric paths to the peer, hosts maintain some state for their receive paths. The possible receive path states are depicted in Figure 3.

```
  v
+--------+    | send first NEW_CONNECTION_ID with the associated Path ID
| ACTIVE |
+--------+    | receive PATH_UPDATE |
| CLOSED |
+--------+
```

Figure 3: Finite-State Machine of receive paths
Upon the sending of the NEW_CONNECTION_ID proposing the corresponding Path ID, the receive path is in ACTIVE state. In this state, hosts MUST ensure that they can associate a packet with the PCID and its corresponding receive path at any time, as they must be ready to figure out the path used by the remote host. ACTIVE is the state where the receive path can be used to receive packets.

Eventually, a receive path may be closed. This is signaled by the reception of a PATH_UPDATE frame. In that case, the path is in CLOSED state. In that state, packets MUST NOT be received over it. A host MUST keep some minimal receive path state to avoid ambiguities and CLOSED path reuse.

In any receive path state (ACTIVE or CLOSED), hosts MUST keep the following receive path information:

- Path ID: encoded as a 8-byte integer. It uniquely identifies the receive path in the connection. This value is immutable.

- Main Connection IDs: they make the link between the path and the QUIC connection it belongs to. These values are immutable.

- Source Path Connection IDs: it makes the link between the packet’s Connection ID field and the receive path. This value can be updated by sending subsequent NEW_CONNECTION_ID frames.

- Receive Path State: the current state of the receive path, one of the values presented in Figure 3.

In the ACTIVE state, the following elements MUST be tracked:

- Packet Number Space: each receive path is associated with its own monotonically increasing packet number space, dedicated for the packet receptions. Packet number considerations described in [I-D.ietf-quic-transport] apply within a given receive path.

- Current 4-tuple: the tuple (sIP, dIP, sport, dport) observed to receive packets over this path. This value is mutable, because it might receive a packet with a different validated remote address and/or port than the one currently recorded. If an host observes a change in the 4-tuple of the receive path, it SHOULD send a NEW_CONNECTION_ID with an increased Retire Prior To field to make the peer change the Path Connection ID.

- Current (local Address ID, remote Address ID) tuple: those identifiers come from the ADD_ADDRESS sent (local) and received
The addresses on which the connection was established have Address ID 0. The reception of PATHS frames helps hosts to associate the remote Address ID used by the path.

- Performance metrics: basic statistics such as number of packets received can be collected on a per-receive path basis.

5.4. Sending Path State

During the Multipath QUIC connection, hosts maintain some state for sending paths. Information about the sending path that hosts are required to store depends on its state. The possible sending path states are depicted in Figure 4.

```
+----------+   | receive a first NEW_CONNECTION_ID with the associated Path ID |
| READY   | v   +----------+
|         |   +----------+
| path usage or PATH_UPDATE |   | reception or sending of REMOVE_ADDRESS |
| v   +----------+   +----------+ 4-tuple validated
| ACTIVE | DISABLED |
| +----------+
| sending PATH_UPDATE |
| v   +----------+
| CLOSED |
| +----------+
```

![Figure 4: Finite-State Machine of sending paths](image)

Once a sending path has been proposed by the peer in a received NEW_CONNECTION_ID frame, it is in the READY state. In this situation, hosts MUST keep the following sending path information:

- Path ID: encoded as a 4-byte integer. It uniquely identifies the sending path in the connection. This value is immutable.

- Main Connection IDs: they make the link between the path and the QUIC connection it belongs to. These values are immutable.
Path Connection IDs: they make the link between the packet’s Connection ID field and the sending path. This value can be updated with subsequent NEW_CONNECTION_ID frames.

Sending Path State: the current state of the sending path, one of the values presented in Figure 4.

When the host wants to start using the path over a validated address, the path goes to the ACTIVE state. This is the state where a sending path can be used to send packets. Having a path in ACTIVE state only guarantees that it can be used, but the host is not forced to. In addition to the fields required in the READY state, the following elements MUST be tracked:

Packet Number Space: each sending path is associated with its own monotonically increasing packet number space, dedicated for the packet sendings. Packet number considerations described in [I-D.ietf-quic-transport] apply within a given sending path.

Current 4-tuple: the tuple (sIP, dIP, sport, dport) used to send packets over this path. This value is mutable, as the host might decide to change its local address and/or port. A host that changes the 4-tuple of a sending path SHOULD migrate it.

Current (local Address ID, remote Address ID) tuple: those identifiers come from the ADD_ADDRESS sent (local) and received (remote). This enables a host to temporarily stop using a sending path when, e.g., the remote Address ID is declared as lost in a REMOVE_ADDRESS. The addresses on which the connection was established have Address ID 0. The reception of PATHS frames helps hosts to associate the remote Address ID used by the path.

Performance metrics: basic statistics such as one-way delay or number of packets sent can be collected on a per-path basis. This information can be useful when a host needs to perform packet scheduling decisions and flow control management.

It might happen that a sending path is temporarily unavailable, because one of the endpoint’s addresses is no more available or because the server decided to decrease the number of active paths. In such cases, the path goes to the DISABLED state. In that state, the host MUST NOT send packets on it. At this state, the host might want to restart using the path over a validated 4-tuple, switching the path state back to ACTIVE state. However, its congestion window MUST be restarted and its performance metrics SHOULD be reset.

Eventually, a sending path may be closed. This is signaled by the PATH_UPDATE frame. In that case, the sending path is in CLOSED state.
state. In that state, packets MUST NOT be sent over it. A host MUST keep the same path state field as in the READY state, to avoid ambiguities and CLOSED path reuse.

5.5. Dynamic Management of Paths

Both hosts determine how many sending paths their peer can use over a Multipath QUIC connection. It is their responsibility to propose asymmetric Path IDs with corresponding PCIDs that could be used by the peer. In addition, they dynamically control the number of active paths that can be used for the connection, as they provide and potentially retire the Connection IDs usable by the peer. A host can propose several different Path Connection IDs with NEW_CONNECTION_ID frames for a given Path ID. This can be useful in migration scenarios.

5.6. Losing Addresses

During the lifetime of a connection, a host might lose addresses, e.g., a network interface that was shut down. All the ACTIVE sending paths that were using that local address MUST stop sending packets from that address. To advertise the address loss to the peer, the host MUST send a REMOVE_ADDRESS frame indicating which Address IDs has been lost. The host MUST also send a PATHS frame indicating the status of the remaining ACTIVE paths.

Upon reception of the REMOVE_ADDRESS, the receiving host MUST stop using the ACTIVE paths affected by the address removal.

Hosts MAY reuse one of these sending paths by changing the assigned 4-tuple. In this case, it MUST send a PATHS frame describing that change.

5.7. Scheduling Strategies

The current QUIC design [I-D.ietf-quic-transport] offers a two-dimensional scheduling space, i.e., which frames will be packed inside a given packet. With the use of multiple paths, a third dimension is added, i.e., the sending path on which the packet will be sent. This dimension can have a non negligible impact on the operations of Multipath QUIC, especially if the available sending paths exhibit very different network characteristics.

The progression of the data flow depends on the reception of the MAX_DATA and MAX_STREAM_DATA frames. Those frames SHOULD be duplicated on several or all paths in use. This would limit the head-of-line blocking issue due to the transmission of the frames over a slow path.
The sending path on which ACK frames are sent impacts the peer. The
ACK frame is notably used to determine the latency of a combination
of asymmetric path. In particular, the peer can compute the round-
trip-time of the combination of its sending path with its receive
one. The peer would compute the latency as the sum of the forward
delay of the acknowledged path and the return delay of the path used
to send the ACK frame. Choosing between acknowledging packets
symmetrically (on path B to A if packet was sent on A to B) or not is
up to the implementation. However, hosts SHOULD keep a consistent
acknowledgement strategy. Selecting a random path to acknowledge
packets will possibly increase the variability of the latency
estimation, especially if paths exhibit very different network
characteristics. Notice that with a receive timestamp field in the
ACK frame, this would enable host to estimate each asymmetric path
one-way delay. Unlike MAX_DATA and MAX_STREAM_DATA, ACK frames
SHOULD NOT be systematically duplicated on several paths as they can
induce a large network overhead.

6. Modifications to QUIC frames

The multipath extension allows hosts to send packets over multiple
paths. Since nearly all QUIC frames are independent of packets, no
change is required for most of them. The only exceptions are the
NEW_CONNECTION_ID and the ACK frames. The NEW_CONNECTION_ID and
RETIRE_CONNECTION_ID are modified to provide Destination Path
Connection ID negotiation for each asymmetric path. The ACK frame
contains packet-level information with the Largest Acknowledged
field. Since the Packet Numbers are now associated to asymmetric
paths, the ACK frame must contain the Path ID it acknowledges.

6.1. NEW_CONNECTION_ID Frame

The NEW_CONNECTION_ID frame (type=0x0b) as defined by
[I-D.ietf-quic-transport] keeps its ability to provide the client
with alternative connection IDs that can be used to break linkability
when migrating connections. It also allows the host to indicate
which destination connection IDs the peer must use to take advantage
of multiple sending paths.

The NEW_CONNECTION_ID is as follows:
Compared to the frame specified in [I-D.ietf-quic-transport], a Path ID field of variable size is prefixed to associate the Path ID with the Connection ID. If the multipath extension was not negotiated during the connection establishment, the NEW_CONNECTION_ID frame is the same as the one presented in [I-D.ietf-quic-transport]. This frame can be sent by both hosts. Upon reception of the frame with a specified path ID, the peer can update the related sending path state either to READY or ACTIVE and store the communicated Connection ID as the Destination Connection ID of the sending path.

A host MUST NOT start using a path as long as the peer have not proposed a Destination Connection ID with a NEW_CONNECTION_ID frame. To limit the delay of the multipath usage upon handshake completion, hosts SHOULD send NEW_CONNECTION_ID frames for receive paths they allow using as soon the connection establishment completes.

To cope with privacy issues, it should be hard to make the link between two different connections or two different paths of a same connection by just looking at the Connection ID contained in packets. Therefore, Path Connection IDs chosen by hosts MUST be random.
6.2. RETIRE_CONNECTION_ID Frame

The RETIRE_CONNECTION_ID frame (type=0x19) as defined by [I-D.ietf-quic-transport] still allows an endpoint to indicate that it will no longer use a connection ID that was issued by its peer. The multipath extensions link Connection IDs to paths. Therefore, this frame should contain the Path ID on which it applies.

The format of the adapted RETIRE_CONNECTION_ID is shown below.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Path ID  (i)                       ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sequence Number (i)                    ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: RETIRE_CONNECTION_ID frame adapted to Multipath QUIC

The frame is handled as described in [I-D.ietf-quic-transport] on a path basis.

6.3. ACK Frame

The format of the modified ACK frame is shown below.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Path ID (i)                        ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Largest Acknowledged (i)                 ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         ACK Delay (i)                      ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      ACK Range Count (i)                    ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      First ACK Range (i)                    ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         ACK Ranges (*)                      ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         [ECN Counts]                        ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: ACK frame adapted to Multipath
Compared to the ACK frame in the current QUIC design [I-D.ietf-quic-transport], the ACK frame is prefixed by a variable size Path ID field indicating to which path the acknowledged PSNs relate to. Notice that if the multipath extension was not negotiated during the connection handshake, the ACK frame is the same as the one presented in [I-D.ietf-quic-transport].

Since frames are independent of packets, and the path notion relates to the packets, the ACK frames can be sent on any path, unlike Multipath TCP [RFC6824] which is constrained to send ACKs on the same path.

Notice that using timestamps would help peers to estimate the one-way delay of the paths. These would also be beneficial for single-path cases.

7. New Frames

To support the multipath operations, new frames have been defined to coordinate hosts. This draft uses a type field containing 0x20 to indicate that the frame is related to multipath operations.

7.1. PATH_UPDATE Frame

The PATH_UPDATE frame is used by a host either to migrate a path or to close it. This indicates to the remote that the closed path MUST NOT be used anymore and it can use the proposed one instead, if any. The proposed type for the PATH_UPDATE is 0x21. A PATH_UPDATE frame is shown below.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Closed Path ID (i)                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Proposed Path ID (i)                  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: PATH_UPDATE Frame

The PATH_UPDATE frame contains the following fields:

- Closed Path ID: A variable-length integer corresponding to the Path ID of the path that is closed.

- Proposed Path ID: A variable-length integer corresponding to the Path ID of the path that substitutes the closed path. If the value is 0, no path is proposed.
Upon the transmission or the reception of the PATH_UPDATE frame, the path with the Path ID referenced in Closed Path ID MUST be in the CLOSED state. If the proposed Path ID is different of 0, the path MUST be ACTIVE.

### 7.2. ADD_ADDRESS Frame

The ADD_ADDRESS frame is used by a host to advertise its currently reachable addresses. The proposed type for the ADD_ADDRESS frame is 0x22. An ADD_ADDRESS frame is shown below.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0|0|P|IPVers.|Address ID (8) |Interface T.(8)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       IP Address (32/128)                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          [Port (16)]          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: ADD_ADDRESS Frame

The ADD_ADDRESS frame contains the following fields:

- **Reserved bits**: the three most-significant bits of the first byte are set to 0, and are reserved for future use.
- **P bit**: the fourth most-significant bit of the first byte indicates, if set, the presence of the Port field.
- **IPVers.**: the remaining four least-significant bits of the first byte contain the version of the IP address contained in the IP Address field.
- **Address ID**: an unique identifier for the advertised address for tracking and removal purposes. This is needed when, e.g., a NAT changes the IP address such that both hosts see different IP addresses for a same path endpoint.
- **Interface Type**: used to provide an indication about the interface type to which this address is bound. The following values are defined: o 0: fixed. Used as default value. o 1: WLAN o 2: cellular
- **IP Address**: the advertised IP address, in network order.
Port: this optional field indicates the port number related to the advertised IP address. When this field is present, it indicates that a path can use the 2-tuple (IP addr, port).

Upon reception of an ADD_ADDRESS frame, the receiver SHOULD store the communicated address for future use. The receiver MUST NOT send packets others than validation ones to the communicated address without having validated it. ADD_ADDRESS frames SHOULD contain globally reachable addresses. Link-local and possibly private addresses SHOULD NOT be exchanged.

### 7.3. REMOVE_ADDRESS Frame

The REMOVE_ADDRESS frame is used by a host to signal that a previously announced address was lost. The proposed type for the REMOVE_ADDRESS frame is 0x23. A REMOVE_ADDRESS frame is shown below.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|Address ID (8) |
+-+-+-+-+-+-+-+-+
```

The frame contains only one field, Address ID, being the identifier of the address to remove. A host SHOULD stop using paths using the removed address and set them in the UNUSABLE state. If the REMOVE_ADDRESS contains an Address ID that was not previously announced, the receiver MUST silently ignore the frame.

### 7.4. PATHS Frame

The PATHS frame communicates the paths state of the sending host to the peer. It allows the sender to communicate its active paths to the peer in order to detect potential connectivity issue over paths. Its proposed type is 0x24. The format of the PATHS frame is shown below.
The PATHS frame contains the following fields:

- **Sequence**: A variable-length integer. This value starts at 0 and increases by 1 for each PATHS frame sent by the host. It allows identifying the most recent PATHS frame.

- **ActiveReceivePaths**: the current number of additional receive paths considered as being active from the sender point of view, i.e., (the number of active receive paths - 1).

- **ActiveSendingPaths**: the current number of additional sending paths considered as being active from the sender point of view, i.e., (the number of active sending paths - 1).

- **Receive Path Info Section**: contains information about all the active receive paths (i.e., there are ActivePaths + 1 entries).

- **Sending Path Info Section**: contains information about all the active sending paths (i.e., there are ActivePaths + 1 entries).

Both Receive Path Info and Sending Path Info Sections share the same format, which is shown below.
The fields in the Path Info Section are:

- **Path ID**: the Path ID of the active path the sending host provides information about.
- **LocAddrID**: the local Address ID of the address currently used by the path.
- **RemAddrID**: the remote Address ID of the address currently used by the path.

The Path Info section only contains the Local and Remote Address IDs so far, but this section can be extended later with other potentially useful information.

### 8. Security Considerations

#### 8.1. Nonce Computation

With Multipath QUIC, each path has its own packet number space. With the current nonce computation [I-D.ietf-quic-tls], using twice the same packet number over two different paths leads to the same cryptographic nonce. Depending on the size of the Initial Value (and hence the nonce), there are two ways to mitigate this issue.

If the Initial Value has a length of 8 bytes, then a packet number used on a given path MUST NOT be reused on another path of the connection, and therefore at most $2^{64}$ packets can be sent on a QUIC connection. This means there will be packet number skipping at path level, but the packet number will remain monotonically increasing on each path.
If the Initial Value has a length of 9 or more, then the cryptographic nonce computation is now performed as follows. The nonce, N, is formed by combining the packet protection IV (either client_pp_iv_n or server_pp_iv_n) with the Path ID and the packet number. The 64 bits of the reconstructed QUIC packet number in network byte order is left-padded with zeros to the size of the IV. The Path ID encoded in its variable-length format is right-padded with zeros to the size of the IV. The Path IV is computed as the exclusive OR of the padded Path ID and the IV. The exclusive OR of the padded packet number and the Path IV forms the AEAD nonce.

8.2. Validation of Exchanged Addresses

To use addresses communicated by the peer through ADD_ADDRESS frames, hosts are required to validate them before using paths to these addresses. The main reason for this validation is that the remote host might have sent, purposely or not, a packet with a source IP that does not correspond to the IP of the remote interface. This could lead to amplification attacks where the client start using a new path with a source IP corresponding to the victim’s one. Without validation, the server might then flood the victim. Similarly for ADD_ADDRESS frames, a malicious server might advertise the IP address of a victim, hoping that the client will use it without validating it before.

9. IANA Considerations

9.1. QUIC Transport Parameter Registry

This document defines a new transport parameter for the negotiation of multiple paths. The following entry in Table 1 should be added to the "QUIC Transport Parameters" registry under the "QUIC Protocol" heading.

<table>
<thead>
<tr>
<th>Value</th>
<th>Parameter Name</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0020</td>
<td>max_paths</td>
<td>Section 5.1.1</td>
</tr>
</tbody>
</table>

Table 1: Addition to QUIC Transport Parameters Entries

10. Acknowledgements

We would like to thanks Masahiro Kozuka and Kazuho Oku for their numerous comments on the first version of this draft. We also thanks Philipp Tiesel for his early comments that led to the current design, and Ian Swett for later feedbacks. We also want to thank Christian
Huitema for his draft about multipath requirements to identify critical elements about the multipath feature. Mohamed Boucadair provided lot of useful inputs on the second version of this document. Maxime Piraux helped us to improve the third version of this draft.

11. References

11.1. Normative References

[I-D.ietf-quic-invariants]

[I-D.ietf-quic-recovery]

[I-D.ietf-quic-tls]

[I-D.ietf-quic-transport]


11.2. Informative References


[I-D.huitema-quic-mpath-req]


Appendix A. Change Log

A.1. Since draft-deconinck-quic-multipath-02
  - Consider asymmetric paths

A.2. Since draft-deconinck-quic-multipath-01
  - Include path policies considerations
  - Add practical considerations thanks to Mohamed Boucadair inputs
  - Adapt the RETIRE_CONNECTION_ID frame
  - Updated text to match draft-ietf-quic-transport-18

A.3. Since draft-deconinck-quic-multipath-00
  - Comply with asymmetric Connection IDs
  - Add max_paths transport parameter to negotiate initial number of active paths
  - Path ID as a regular varint
  - Remove max_path_id transport parameter
  - Updated text to match draft-ietf-quic-transport-14

A.4. Since draft-deconinck-multipath-quic-00
  - Added PATH_UPDATE frame
  - Added MAX_PATHS frame
  - No more packet header change
  - Implicit Path ID notification using Connection ID and NEW_CONNECTION_ID frames
  - Variable-length encoding for Path ID
  - Updated text to match draft-ietf-quic-transport-10
  - Fixed various typos
Authors’ Addresses

Quentin De Coninck
UCLouvain

Email: quentin.deconinck@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: Olivier.Bonaventure@uclouvain.be