QUIC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 4 November 2021

Q. De Coninck
O. Bonaventure
UCLouvain

3 May 2021

Multipath Extensions for QUIC (MP-QUIC)
draft-deconinck-quic-multipath-07

Abstract

This document specifies extensions to the QUIC protocol to enable the simultaneous usage of multiple paths for a single connection. These extensions are compliant with the single-path QUIC design and preserve QUIC privacy features.

Discussion about this draft is encouraged either on the QUIC IETF mailing list quic@ietf.org or on the GitHub repository which contains the draft: https://github.com/qdeconinck/draft-deconinck-multipath-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 4 November 2021.

Copyright Notice

Table of Contents

1. Introduction

Today's endhosts are equipped with several network interfaces. Users expect to be able to seamlessly switch from one interface to another one or use them simultaneously to aggregate bandwidth whenever needed. During the last years, several multipath extensions to transport protocols have been proposed (e.g., [RFC8684], [MPRTP], or [SCTPCMT]). Multipath TCP [RFC8684] is the most mature one. It is already deployed on popular smartphones, but also for other use cases [RFC8041] [IETFJ].

With regular TCP and UDP, all the packets that belong to a given flow share the same 4-tuple {source IP address, source port number, destination IP address, destination port number} that acts as an identifier for this flow. This prevents these flows from using multiple paths.

QUIC [I-D.ietf-quic-transport] does not use the 4-tuple as an implicit connection identifier. Instead, a QUIC flow is identified by a Connection ID. This enables QUIC to cope with events affecting the 4-tuple, such as NAT rebinding or IP address changes. The QUIC connection migration feature, specified in Section 9 of [I-D.ietf-quic-transport], enables migrating a flow from one 4-tuple to another one, sustaining in particular a connection over different network paths.

Still, there is a void to specify simultaneous usage of QUIC over available network 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 [RFC8041][IETFJ]. Experience with Multipath TCP on smartphones shows that the ability to simultaneously use WLAN and cellular during

handovers improves the user perceived quality of experience. A performance evaluation of an early solution for such use cases and a comparison between Multipath QUIC and Multipath TCP may be found in [MPQUIC]. A recent publication [MFQUIC] shows how the design presented in this draft enables implementations to take advantage of highly asymmetrical network paths such as satellite links.

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 document to propose extensions to the current QUIC design to enable it to simultaneously use several network paths. This document focuses mainly on network paths that are distinguishable by the endpoints.

This document is organized as follows. It first provides in Section 3 an overview of the operation of Multipath QUIC. It then states in Section 4 how Connection IDs can map to different unidirectional flows (called uniflows) in use. Section 5 specifies the required changes in the current QUIC design [I-D.ietf-quic-transport] to enable the simultaneous usage of multiple network paths. These extensions introduce new frames that are described in Section 6 and Section 7. Finally, Section 8 discusses some security considerations.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

* Uniflow: A unidirectional flow of packets between a QUIC host and its peer. This flow is identified by an internal identifier, called Uniflow ID. Packets sent over a uniflow use a Destination Connection ID that may change during the lifetime of the connection. When being in use, an uniflow is temporarily bound to a 4-tuple (Source IP Address, Source Port Number, Destination IP Address, Destination Port Number).

* Initial Uniflows: The two uniflows used by peers for the establishment of a QUIC connection. One is the uniflow from the client to the server and the other is the uniflow from the server to the client. The cryptographic handshake is done on these uniflows. These are identified by Uniflow ID 0.

2.1. Notation Conventions

Packet and frame diagrams use the format described in Section 12 of [I-D.ietf-quic-transport].

3. Overview

The design of QUIC [I-D.ietf-quic-transport] provides reliable transport with multiplexing, confidentiality, integrity, and authenticity of data flows. 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.

The current design of QUIC does not enable multihomed devices to efficiently use different paths simultaneously. This document proposes multipath extensions with the following design goals:

* Each host keeps control on the number of uniflows being used over the connection.

* The simultaneous usage of multiple uniflows should not introduce new privacy concerns.

* A host must ensure that all the paths it uses actually reach its peer to avoid packet flooding towards a victim (see Section 21.12.3 of [I-D.ietf-quic-transport])

* The multipath extensions should handle the asymmetrical nature of paths between two peers.

We first explain why a multipath extension would be beneficial to QUIC and then describe it at a high level.

3.1. Moving from Bidirectional Paths to Uniflows

To understand the overall architecture of the multipath extensions, let us first refine the notion of "path". A path may be denoted by a 4-tuple (Source IP Address, Source Port Number, Destination IP Address, Destination Port Number). In QUIC, this is namely a UDP path 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 WLAN: one over IPv4 and the other one over IPv6.

A regular QUIC connection is composed of two independent active packet flows. The first flow gathers the packets from the client to the server and the other the packets from the server to the client. To illustrate this, let us consider the example depicted in Figure 1. In this example, the client has two IP addresses: IPc1 and IPc2. The server has one single address: IPs1.

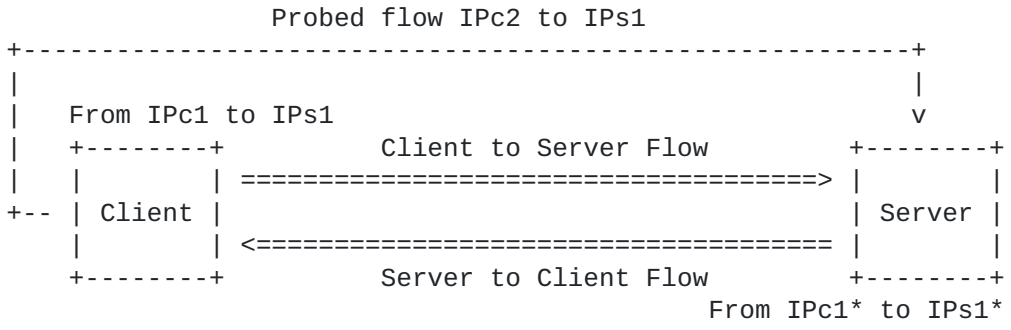


Figure 1: Identifying Unidirectional Flows in QUIC

The client initiates the QUIC connection by sending packets towards the server. The server then replies to the client if it accepts the connection. If the handshake succeeds, the connection is established. Still, this "path" actually consists in two independent UDP flows. Each host has its own view of (i) the 4-tuple used to send packets and (ii) the 4-tuple on which it receives packets. While the 4-tuple used by the client to send packets may be the same as the one seen and used by the server, this is not always the case since middleboxes (e.g., NATs) may alter the 4-tuple of packets.

To further emphasize on this flow asymmetry, QUIC embeds a path validation mechanism [I-D.ietf-quic-transport] assessing whether a host can reach its peer through a given 4-tuple. This process is unidirectional, i.e., the sender checks that it can reach the receiver, but not the reverse. A host receiving a PATH_CHALLENGE frame on a new 4-tuple may in turn initiate a path validation, but this is up to the peer.

A QUIC connection is a collection of unidirectional flows (called, uniflows). A plain QUIC connection is composed of a main uniflow from client to server and another main uniflow from server to client. These uniflows have their own Connection IDs. They are host-specific, i.e., the uniflow(s) from A to B are different from the

ones from B to A. This potentially enables the use of unidirectional links such as non-broadcast satellite links [RFC3077], which cannot be used with 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 (set of) Connection ID(s), 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 smartphone losing the WLAN connectivity can then continue the connection over its cellular interface, for instance.

A QUIC connection can thus start on a given set of uniflows, denoted as the initial uniflows, and end on another ones. However, the current QUIC design [I-D.ietf-quic-transport] assumes that only one pair of uniflows is in use for a given connection. The current transport specification [I-D.ietf-quic-transport] does not provide means to distinguish path migration from simultaneous usage of available uniflows for a given connection.

This document fills that void. It first proposes mechanisms to communicate endhost addresses to the peer. It then leverages the Address Validation procedure with PATH_CHALLENGE and PATH_RESPONSE frames described in Section 8 of [I-D.ietf-quic-transport] to verify whether the additional addresses advertised by the host are reachable. In this case, those addresses can be used to initiate new uniflows to spread packets over several network paths following a packet scheduling policy that is out of scope of this document.

The example of Figure 2 illustrates a data exchange between a dual-homed client sending a request spanning two packets and a single-homed server. Uniflow IDs are independently chosen by each host. In the presented example, the client sends packets over WLAN on Uniflow 0 and over LTE on Uniflow 1, while the packets sent by the server over WLAN are on Uniflow 2 and those over LTE are on Uniflow 1.

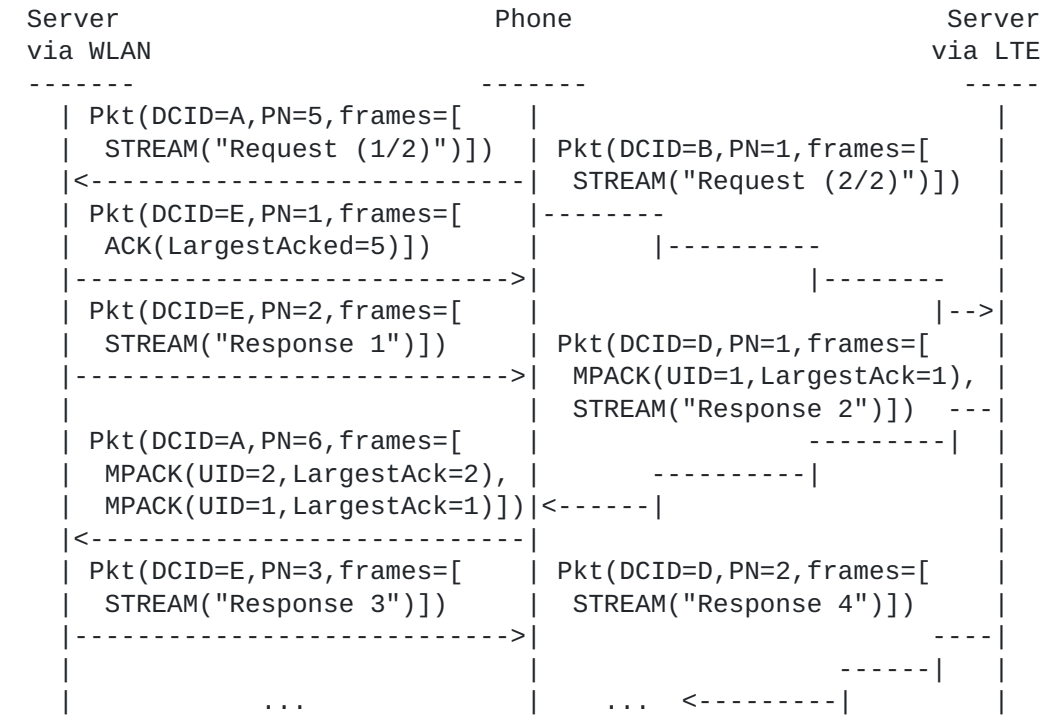


Figure 2: Data flow with Multipath QUIC

The remaining of this section presents a high-level overview of the multipath operations in QUIC.

3.3. Establishment of a Multipath QUIC Connection

A Multipath QUIC connection starts as a regular QUIC connection [I-D.ietf-quic-transport] [I-D.ietf-quic-tls]. The multipath extensions defined in this document are negotiated using the "max_sending_uniflow_id" transport parameter. Any value for this transport parameter advertises the support of the multipath extensions.

When negotiating the multipath extensions, the host puts a upper bound on the number of sending uniflows that it will use over the connection. For instance, if a host wants to spread connection's packets on at most two network paths, it advertises a "max_sending_uniflow_id" of 1. Note that the creation of a second sending uniflow will depend on the peer, as described later.

Notice that a host advertising a value of 0 for the "max_sending_uniflow_id" transport parameter indicates that it does not want additional uniflows to send packets, but it still supports the multipath extensions. Such situation might be useful when the host does not require multiple uniflows for packet sending but still wants to let the peer use multiple uniflows to reach it.

3.4. Architecture of Multipath QUIC

To illustrate the architecture of a Multipath QUIC connection, consider Figure 3.

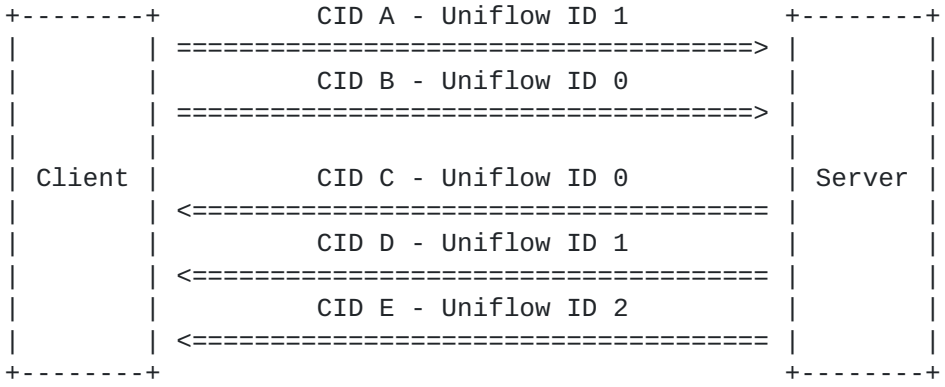


Figure 3: An Example of Uniflow Distribution over a Multipath QUIC Connection

Once established, a Multipath QUIC connection consists in one or more uniflows from the client to the server and one or more uniflows from the server to the client. The number of uniflows in one direction can be different from the one in the other direction. The example in Figure 3 shows two uniflows from the client to the server and three uniflows from the server to the client. From the end-hosts' viewpoint, they observe two kinds of uniflows:

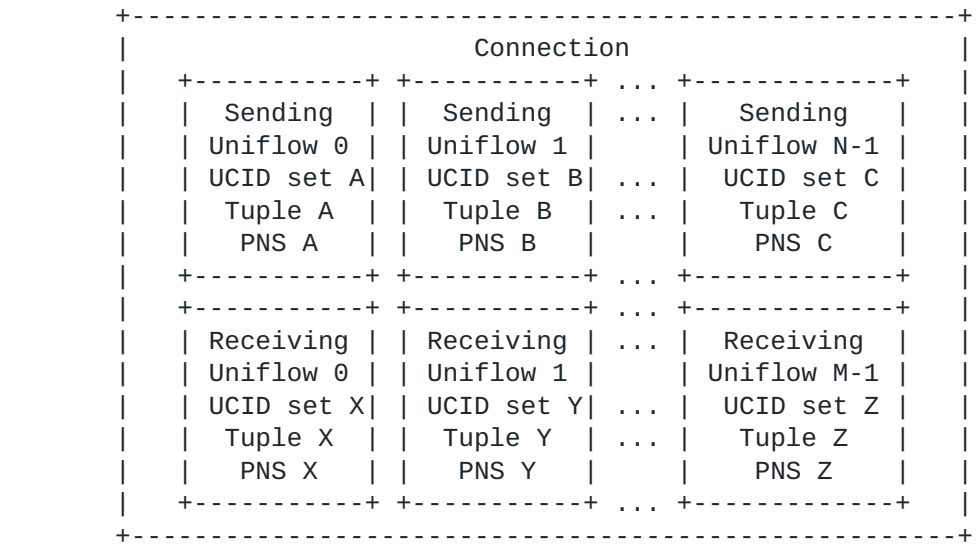
* Sending uniflows: uniflows over which the host can send packets

* Receiving uniflows: uniflows over which the host can receive packets

Reconsidering the example in Figure 3, the client has two sending uniflows and three receiving uniflows. The server has three sending uniflows and two receiving uniflows. There is thus a one-to-one mapping between the sending uniflows of a host and the receiving uniflows of its peer. A uniflow is seen as a sending uniflow from the sender's perspective and as a receiving uniflow from the receiver's viewpoint.

Each uniflow is associated with a specific four-tuple and identified by a Uniflow ID, as shown in Figure 4.

Client state



Server state

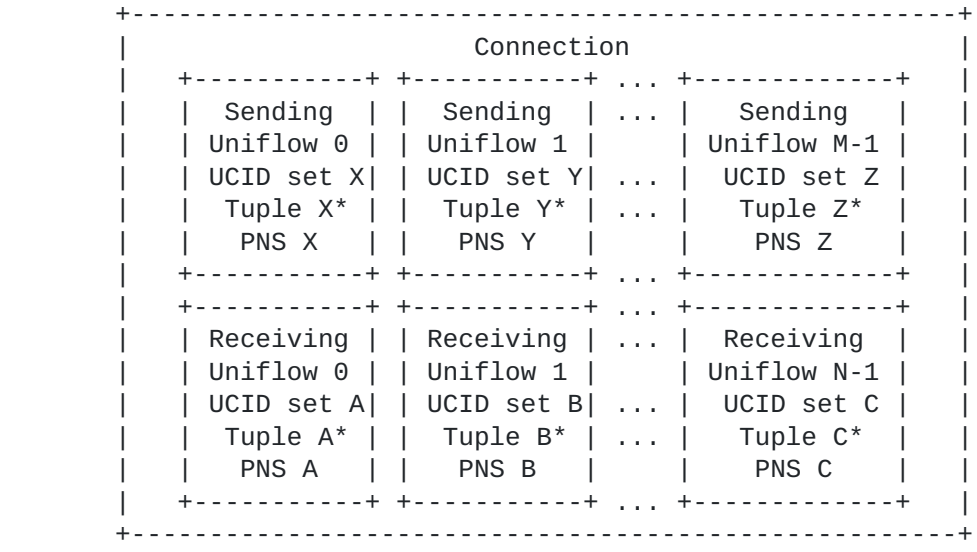


Figure 4: Architectural view of Multipath QUIC for a host having N sending uniflows and M receiving uniflows

A Multipath QUIC connection starts using two Initial Uniflows, identified by Uniflow ID 0 on each peer. The packets can then be spread over several uniflows. Each uniflow has its (set of) Uniflow Connection ID(s) (UCID) packets that are used to explicitly mark

where they belong to. Depending on the direction of the uniflow, the host keeps either the Uniflow Source Connection ID (USCID, for the receiving uniflows) or the Uniflow Destination Connection ID (USCID, for the sending uniflows). Notice that the (set of) UDCID(s) of a sending uniflow of a host is the same as the (set of) USCID(s) of the corresponding receive uniflow of the remote peer.

Preventing the linkability of different uniflows is an important requirement for the multipath extensions [I-D.huitema-quic-mpath-req]. We address it by using UCIDs as implicit uniflow identifiers. This makes the linkability harder than having explicit signaling as in earlier version of this draft. Furthermore, it does not require any public header change and thus preserves the QUIC invariants [I-D.ietf-quic-invariants].

When a uniflow is in use, each endhost associates it with a network path. In practice, this consists in a particular 4-tuple over which packets are sent (or received) on a sending (or receiving) uniflow. Each endhost has a specific vision of the 4-tuple, which might differ between endhosts. For instance, a client located behind a NAT sends data from a private IP address and the server will receive packets coming from the NAT's public IP address. Notice that while uniflows may share a common network path, this is not mandatory.

Each uniflow is an independent flow of packets over a given network path. Uniflows can experience very different network conditions (latency, bandwidth, ...). To handle this, each uniflow has its own packet sequence number space.

In addition to the UCIDs, 4-tuple and packet number space, some additional information is maintained for each uniflow. The Uniflow ID identifies the uniflow at the frame level and ensures uniqueness of the nonce (see Section 8.1 for details) while limiting the number of concurrently used uniflows.

3.5. Uniflow Establishment

The "max_sending_uniflow_id" transport parameter exchanged during the cryptographic handshake fixes an upper bound on the number of sending uniflows a host wants to support. Then, hosts provide to their peer Uniflow Connection IDs to use on uniflows. Both hosts dynamically control how many sending uniflows can currently be in use by the peer, i.e., the number of different Uniflow IDs proposed to the peer. While the sender determines the upper bound of sending paths it can have, it is the receiver that initializes uniflows, as the sender needs a UCID communicated by the receiver before using a uniflow.

Notice that the peers might advertise different values for the "max_sending_uniflow_id" transport parameters, setting different upper bounds to the sending and receiving uniflows of each host.

Hosts initiate the creation of their receiving uniflows by sending MP_NEW_CONNECTION_ID frames (see Section 6.1) which are an extended version of the NEW_CONNECTION_ID frame. This frame associates a UCID to a uniflow. Upon reception of the MP_NEW_CONNECTION_ID frame, a host can start using the proposed sending uniflow having the referenced Uniflow ID by marking sent packets with the provided UCID. Therefore, once a host sends a MP_NEW_CONNECTION_ID frame, it announces that it is ready to receive packets from that Uniflow ID with the proposed UCID. As frames are encrypted, adding new uniflows over a QUIC connection does not leak cleartext identifiers [I-D.huitema-quic-mpath-req].

A server might provide several Uniflow Connection IDs for the same Uniflow ID with multiple MP_NEW_CONNECTION_ID frames. This can be useful to cope with migration cases, as described in Section 3.9. Multipath QUIC is thus asymmetrical.

3.6. Exchanging Data over Multiple Uniflows

A QUIC packet acts as a container for one or more frames. Multipath QUIC uses the same STREAM frames as QUIC to carry data. A byte offset is associated with the data payload. One of the key design of (Multipath) QUIC is that frames are independent of the packets carrying them. This implies that a frame transmitted over one uniflow could be retransmitted later on another uniflow without any change. Furthermore, all current QUIC frames are idempotent and could be optimistically duplicated over several uniflows.

The uniflow on which data is sent is a packet-level information. This means that a frame can be sent regardless of the uniflow of the packet carrying it. Other flow control considerations like the stream receive window advertised by the MAX_STREAM_DATA frame remain unchanged when there are multiple sending uniflows.

As previously described, Multipath QUIC might face reordering at packet-level when using uniflows having different latencies. The presence of different Uniflow Connection IDs ensures that the packets sent over a given uniflow will contain monotonically increasing packet numbers. To ensure more flexibility and potentially to reduce the ACK block section of the (MP_)ACK frame when aggregating bandwidth of uniflows exhibiting different network characteristics, each uniflow keeps its own monotonically increasing Packet Number space. This potentially allows sending up to 2 * 2^64 packets on a QUIC connection since each uniflow has its own packet number space (see Section 8.1 for the detail of this limit).

With the introduction of multiple uniflows, there is a need to acknowledge packets sent on different uniflows separately. The packets sent on Initial Uniflows (with Uniflow ID 0) are still acknowledged with regular ACK frames, such that no modification is introduced in a core frame. For the other uniflows, the multipath extensions introduce a MP_ACK frame which prefixes the ACK frame with a Uniflow ID field indicating from which receiving uniflow the host acknowledges packets. To better explain this, let us consider the situation illustrated in Figure 5.

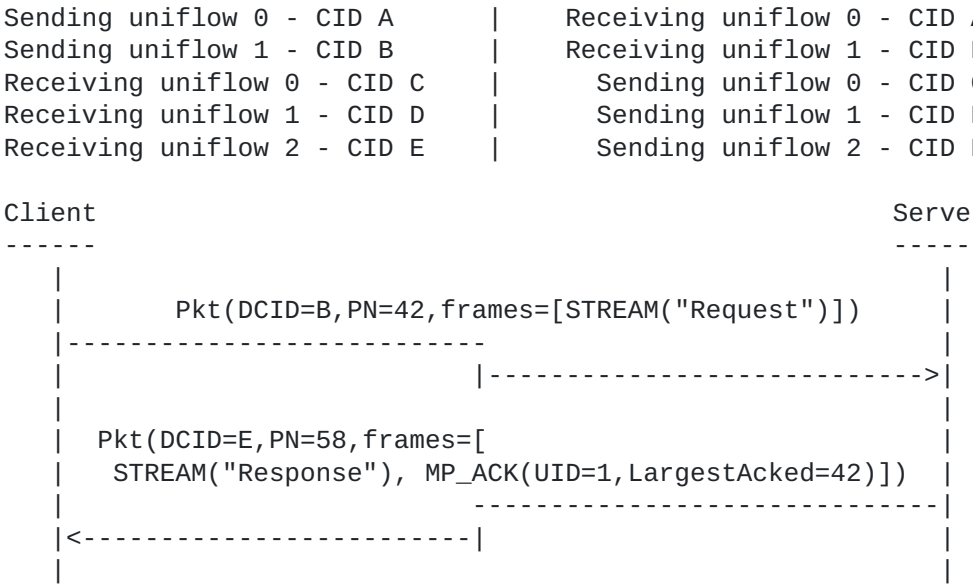


Figure 5: Acknowledging Packets Sent on Uniflows

Here five uniflows are in use, two in the client to server direction and three in the reverse one. The client first sends a packet on its sending Uniflow 1 (linked to CID B). The server receives the packet on its receiving Uniflow 1. Therefore, it generates a MP_ACK frame for Uniflow ID 1 and transmits it to the client. The server can

choose any of its sending uniflows to transmit this frame. In the provided situation, the server sends its packets on Uniflow 2. The client thus receives this packet on its receiving Uniflow 2.

Similarly, packets sent over a given uniflow might be acknowledged by (MP_)ACK frames sent on another uniflow that does not share the same network path. Looking at Figure 2 again, "Response 2" packet on server's sending uniflow 1 with DCID D using the LTE network is acknowledged by a MP_ACK frame received on a uniflow using the WLAN network.

3.7. Exchanging Addresses

When a multi-homed device connects to a dual-stacked server using its IPv4 address, it is aware of its local addresses (e.g., the WLAN and the cellular ones) and the IPv4 remote address used to establish the QUIC connection. If the client wants to create new uniflows and use them over the IPv6 network, 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 MUST first be validated by the receiving host before it starts using it as described in Section 8 of [I-D.ietf-quic-transport]. This process ensures that the advertised address actually belongs to the peer and that the peer can receive packets sent by the host on the provided 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 using its sending uniflows. To enable the server to make the link between the private and the public addresses and hence conciliate the different 4-tuple views, Multipath QUIC provides the UNIFLOWS frame that lists the current active sending Uniflow IDs along with their associated local Address ID. Notice that a host might also discover the public addresses of its peer by observing its remote IP addresses associated to the connection.

A receiving uniflow is active as soon as the host has sent the MP_NEW_CONNECTION_ID frames proposing the corresponding Uniflow Connection IDs to its peer. A sending uniflow is active when it has received its Uniflow Connection IDs and is bound to a validated 4-tuple. The UNIFLOWS frame indicates the local Address IDs that the uniflow uses from the sender's perspective. With this information, the remote host can validate the public address and associate the advertised one with the perceived addresses.

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 reach of a WLAN network or shutting off one of its network interfaces. Such address removals are advertised using REMOVE_ADDRESS frames. The REMOVE_ADDRESS frame contains the Address ID of the lost address previously communicated through ADD_ADDRESS. Notice that because a given Address ID might encounter several events that need to be ordered (e.g., ADD_ADDRESS, REMOVE_ADDRESS and ADD_ADDRESS again), both ADD_ADDRESS and REMOVE_ADDRESS frames include an Address ID related Sequence Number.

3.9. Uniflow Migration

At a given time, a Multipath QUIC endpoint gathers a set of active sending and receiving uniflows, each associated to a 4-tuple. This association is mutable. Hosts can change the 4-tuple used by their sending uniflows at any time, enabling QUIC to migrate uniflows from one network path to another. Yet, to address privacy issues due to the linkability of addresses, hosts should avoid reusing the same Connection ID used by a sending uniflow when the 4-tuple changes, as described in Section 9.5 of [I-D.ietf-quic-transport].

3.10. Handling Multiple Network Paths

The simultaneous usage of several sending uniflows introduces new algorithms (packet scheduling, path management) whose specifications are out of scope of this document. Nevertheless, these algorithms are actually present in any multipath-enabled transport protocol like Multipath TCP, CMT-SCTP and Multipath DCCP. A companion draft [I-D.bonaventure-iccrg-schedulers] provides several general-purpose packet schedulers depending on the application goals. A similar document can be created to discuss path/uniflow management considerations.

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 uniflows are active. Using the congestion control scheme defined in [I-D.ietf-quic-recovery] with Multipath QUIC would result in unfairness. Each sending uniflow of a Multipath QUIC connection MUST have its own congestion control state. As for Multipath TCP, the windows of the different sending uniflows MUST be coupled together [RFC6356].

4. Mapping Uniflow IDs to Connection IDs

As described in the overview section, hosts need to identify on which uniflows packets are sent. The Uniflow ID must then be inferred from the public header. This is done by using the Destination Connection ID field of Short Header packets.

The Initial Uniflow Connection IDs are determined during the cryptographic handshake and actually correspond to both Connection IDs in the current single-path QUIC design [I-D.ietf-quic-transport]. Additional Uniflow Connection IDs for the Initial Uniflows can be provided with the regular NEW_CONNECTION_ID frames. The Uniflow Connection IDs of the other uniflows are determined when the MP_NEW_CONNECTION_ID frames are exchanged.

Hosts MUST accept packets coming from their peer using the UCIDs they proposed in the (MP_)NEW_CONNECTION_ID frames they sent and associate them with the corresponding receiving Uniflow ID. Upon reception of a (MP_)NEW_CONNECTION_ID frame, hosts MUST acknowledge it and MUST store the advertised Uniflow Destination Connection ID and the Uniflow ID of the proposed sending uniflow.

Hosts MUST ensure that all advertised Uniflow Connection IDs are available for the whole connection lifetime, unless they have been retired by their peer in the meantime by the reception of a (MP_)RETIRE_CONNECTION_ID.

A host MUST NOT send MP_NEW_CONNECTION_ID frames with a Uniflow ID greater than the value of "max_sending_uniflow_id" advertised by its peer.

5. Using Multiple Uniflows

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_uniflow_id" transport parameter. A QUIC connection is initially single-path in QUIC. During this handshake, hosts advertise their support for multipath operations. When a host advertises a value for the "max_sending_uniflow_id" transport parameter, it indicates that it supports the multipath extensions, i.e., the extensions defined in this document (not to be mixed with the availability of local multiple network paths). If any host does not advertise the "max_sending_uniflow_id" transport parameter, multipath extensions are disabled.

The usage of multiple uniflows relies on the ability to use several Connection IDs over a same QUIC connection. Therefore, zero-length Connection IDs MUST NOT be used if the peer advertises a value different from 0 for the "max_sending_uniflow_id" transport parameter.

5.1.1. Transport Parameter Definition

A host MAY use the following transport parameter:

max_sending_uniflow_id (0x40): Indicates the support of the multipath extension presented in this document, regardless of the carried value. Its integer value puts an upper bound on the number of sending uniflows the host advertising the value is ready to support. If absent, this means that the host does not agree to use the multipath extension over the connection.

5.2. Coping with Additional Remote Addresses

Hosts can learn remote addresses either by receiving ADD_ADDRESS frames or observing the 4-tuple of incoming packets. Hosts MUST first validate the newly learned remote IP addresses before starting sending packets to those addresses. This requirement is explained in Section 8.2. Hosts MUST initiate Address Validation Procedure as specified in [I-D.ietf-quic-transport].

A host MAY cache a validated address for a limited amount of time.

5.3. Receiving Uniflow State

When proposing uniflows to their peer, hosts need to maintain some state for their receiving uniflows. This state is created upon the sending of a first MP_NEW_CONNECTION_ID frame proposing the corresponding Uniflow ID. As long as there is still one active Uniflow Connection ID for this receiving uniflow (i.e., one UCID which was not retired yet using a MP_RETIRE_CONNECTION_ID), the host MUST accept packets over the receiving uniflow. Once created, hosts MUST keep the following receiving uniflow information:

Uniflow ID: An integer that uniquely identifies the receiving uniflow in the connection. This value is immutable.

Uniflow Connection IDs: Possible values for the Connection ID field of packets belonging to this receiving uniflow. This value contains the sequence of active UCIDs that were advertised in previously sent MP_NEW_CONNECTION_ID frames. Notice that this sequence might be empty, e.g., when all advertised UCIDs have been retired by the peer.

Packet Number Space: Packet number space dedicated to this receiving uniflow. Packet number considerations described in Section 12.3 of [I-D.ietf-quic-transport] apply within a given receiving uniflow.

Associated 4-tuple: The tuple (sIP, dIP, sport, dport) currently observed to receive packets over this uniflow. This value is mutable, because a host might receive a packet with a different (possibly) validated remote address and/or port than the one previously recorded. If a host observes a change in the 4-tuple of the receiving uniflow, it follows the considerations of Section 9.5 of [I-D.ietf-quic-transport].

Associated local Address ID: The Address ID advertised in ADD_ADDRESS frames sent by the peer corresponding to the local address used to receive packets. This helps to generate UNIFLOWS frames advertising the mapping between uniflows and addresses. The addresses on which the connection was established have Address ID 0.

Hosts can also collect network measurements on a per-uniflow basis, like the number of packets received.

5.4. Sending Uniflow State

During a Multipath QUIC connection, hosts maintain some state for sending uniflows. The state of the sending uniflow determines information that hosts are required to store. The possible sending uniflow states are depicted in Figure 6.

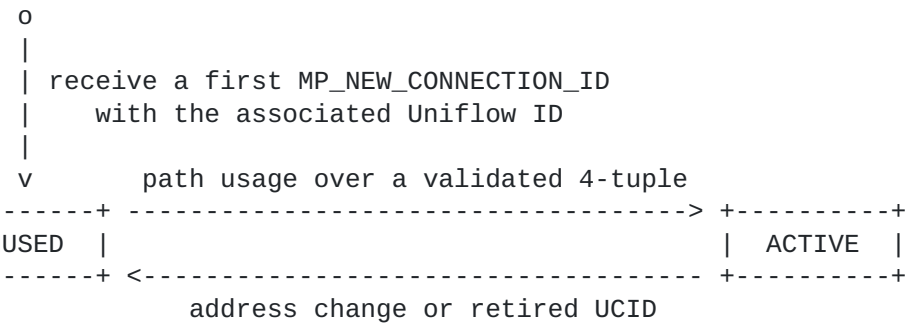


Figure 6: Finite-State Machine describing the possible states of a sending uniflow

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

Uniflow ID: An integer that uniquely identifies the sending uniflow in the connection. This value is immutable.

Uniflow Connection IDs: Possible values for the Connection ID field of packets belonging to this sending uniflow. This value contains the sequence of active UCIDs that were advertised in previously received MP_NEW_CONNECTION_ID frames. Notice that this sequence might be empty, e.g., when all advertised UCIDs have been retired.

Sending Uniflow State: The current state of the sending uniflow, being one of the values presented in Figure 6.

Packet Number Space: Packet number space dedicated to this sending uniflow. Packet number considerations described in Section 12.3 of [I-D.ietf-quic-transport] apply within a given sending uniflow.

Notice that a UNUSED sending uniflow MAY send probing packets to validate a given 4-tuple.

When the host wants to start using the sending uniflow over a validated address, the sending uniflow goes to the ACTIVE state. This is the state where a sending uniflow can be used to send packets. Having an uniflow 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 UNUSED state, the following elements MUST be tracked:

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

Associated (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 uniflow 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 UNIFLOWS frames helps hosts associate the remote Address ID used by the sending uniflow.

Congestion controller: A congestion window limiting the transmission rate of the sending uniflow.

Performance metrics: Basic statistics such as one-way delay or the number of packets sent. This information can be useful when a host needs to perform packet scheduling decisions and flow control.

It might happen that a sending path is temporarily unavailable, because one of the endpoint's addresses is no more available or because the host retired all the UCIDs of the sending uniflow. In such cases, the path goes back to the UNUSED state. When performing a transition back to the UNUSED state, hosts MUST reset the additional state added by the ACTIVE state. In the UNUSED state, the host MUST NOT send non-probing packets on it. At this state, the host might want to restart using the uniflow over another validated 4-tuple, switching the uniflow state back to the ACTIVE state. However, its congestion controller state MUST be restarted and its performance metrics SHOULD be reset.

5.5. 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 uniflows that were using that local address MUST stop sending packets from that address. To advertise the loss of an address to the peer, the host MUST send a REMOVE_ADDRESS frame indicating which local Address IDs has been lost. The host MUST also send an UNIFLOWS frame indicating the status of the remaining ACTIVE uniflows.

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

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

6. New Frames

To support the multipath operations, new frames have been defined to coordinate hosts. The following table summarizes the added frames.

Type Value	Frame Type Name	Definition	Pkts	Spec
0x40	MP_NEW_CONNECTION_ID	Section 6.1	___1	P
0x41	MP_RETIRE_CONNECTION_ID	Section 6.2	___1	
0x42 - 0x43	MP_ACK	Section 6.3	___1	NC
0x44	ADD_ADDRESS	Section 6.4	___1	
0x45	REMOVE_ADDRESS	Section 6.5	___1	
0x46	UNIFLOWS	Section 6.6	___1	

Table 1: Multipath-related Frame Types

Table 1 uses the same notation convention as the Table 3 of [I-D.ietf-quic-transport] for the Pkts and Spec columns. In particular, all frames defined in this document MUST be exchanged in 1-RTT packets. A host receiving one of the multipath-related frames in other encryption context MUST close the connection with a PROTOCOL_VIOLATION error. All the frames are ack-eliciting except the MP_ACK frame. The MP_ACK frame does not count towards bytes in flight if the packet containing it only carries either ACK or MP_ACK frames. The MP_NEW_CONNECTION_ID frame is the only new frame that can be sent to probe new network paths.

The remaining of this document uses the notation convention described in [I-D.ietf-quic-transport].

6.1. MP_NEW_CONNECTION_ID Frames

The MP_NEW_CONNECTION_ID frame (type=0x40) is an extension of the NEW_CONNECTION_ID frame defined by [I-D.ietf-quic-transport]. It provides the peer with alternative Connection IDs and associates them to a particular uniflow using the Uniflow ID.

The format of the MP_NEW_CONNECTION_ID frame is as follows.

```
MP_NEW_CONNECTION_ID Frame {
  Type (i) = 0x40,
  Uniflow ID (i),
  Sequence Number (i),
  Retire Prior To (i),
  Length (8),
  Connection ID (8..160),
  Stateless Reset Token (128),
}
```

Figure 7: MP_NEW_CONNECTION_ID Frame Format

Compared to the NEW_CONNECTION_ID frame specified in [I-D.ietf-quic-transport], the following field is added.

Uniflow ID: Indicates to which uniflow the provided Connection ID relates.

The remaining fields keep the same semantic as for the NEW_CONNECTION_ID frame.

This frame can be sent by both hosts. Upon reception of the frame with a specified Uniflow ID, the peer MUST update the related sending uniflow state and store the communicated Connection ID.

To limit the delay of the multipath usage upon handshake completion, hosts SHOULD send MP_NEW_CONNECTION_ID frames for receive uniflows they allow using as soon the connection establishment completes.

The generation of Connection ID MUST follow the same considerations as presented in Section 5.1 of [I-D.ietf-quic-transport].

6.2. MP_RETIRE_CONNECTION_ID Frame

The MP_RETIRE_CONNECTION_ID frame (type=0x41) is an extension of the RETIRE_CONNECTION_ID frame defined by [I-D.ietf-quic-transport]. It indicates that the end-host will no longer use a Connection ID related to a given uniflow that was issued by its peer.

The format of the MP_RETIRE_CONNECTION_ID frame is shown below.

```
MP_RETIRE_CONNECTION_ID Frame {
  Type (i) = 0x41,
  Uniflow ID (i),
  Sequence Number (i),
}
```

Figure 8: MP_RETIRE_CONNECTION_ID Frame Format

Compared to the RETIRE_CONNECTION_ID frame specified in [I-D.ietf-quic-transport], the following field is added.

Uniflow ID: Indicates on which uniflow the Connection ID is retired.

The frame is handled as the RETIRE_CONNECTION_ID frame described in [I-D.ietf-quic-transport] on an uniflow basis.

6.3. MP_ACK Frame

The MP_ACK frame (types 0x42 and 0x43) is an extension of the ACK frame defined by [I-D.ietf-quic-transport]. It allows hosts to acknowledge packets that were sent on non-initial uniflows. If the frame type is 0x43, MP_ACK frames also contain the sum of QUIC packets with associated ECN marks received on the connection up to this point.

The format of the MP_ACK frame is shown below.

```
MP_ACK Frame {
  Type (i) = 0x02..0x03,
  Uniflow ID (i),
  Largest Acknowledged (i),
  ACK Delay (i),
  ACK Range Count (i),
  First ACK Range (i),
  ACK Range (..) ...,
  [ECN Counts (..)],
}
```

Figure 9: MP_ACK Frame Format

Compared to the ACK frame specified in [I-D.ietf-quic-transport], the following field is added.

Uniflow ID: Indicates on which uniflow the acknowledged packet sequence numbers relate.

6.4. ADD_ADDRESS Frame

The ADD_ADDRESS frame (type=0x44) is used by a host to advertise its currently reachable addresses.

The format of the ADD_ADDRESS frame is shown below.

```
ADD_ADDRESS Frame {
  Type (i) = 0x44,
  Rsv (3) = 0,
  P (1),
  IP Version (4),
  Address ID (8),
  Sequence Number (i),
  Interface Type (8),
  IP Address (32/128),
  [Port (16)],
}
```

Figure 10: ADD_ADDRESS Frame Format

The ADD_ADDRESS frame contains the following fields.

Rsv: These bits are set to 0, and are reserved for future use.

P bit: This bit indicates, if set, the presence of the Port field.

IP Version: Written on 4 bits, contains the version of the IP address contained in the IP Address field.

Address ID: An one-byte 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 network path.

Sequence Number: An Address ID related sequence number of the event, encoded as a variable-length integer. The sequence number space is shared with REMOVE_ADDRESS frames mentioning the same Address ID.

Interface Type: A one-byte field providing an indication about the interface type to which this address is bound. The following values are defined:

* 0: fixed. Used as default value.

* 1: WLAN

* 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 an uniflow 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 as specified in Section 8 of [I-D.ietf-quic-transport]. ADD_ADDRESS frames SHOULD contain globally reachable addresses. Link-local and possibly private addresses SHOULD NOT be exchanged.

6.5. REMOVE_ADDRESS Frame

The REMOVE_ADDRESS frame (type=0x45) is used by a host to signal that a previously announced address was lost.

The format of the REMOVE_ADDRESS frame is shown below.

```
REMOVE_ADDRESS Frame {
  Type (i) = 0x45,
  Address ID (8),
  Sequence Number (i),
}
```

Figure 11: REMOVE_ADDRESS Frame Format

The REMOVE_ADDRESS frame contains the following fields.

Address ID: The one-byte identifier of the address to remove.

Sequence Number: An Address ID related sequence number of the event, encoded as a variable-length integer. The sequence number space is shared with ADD_ADDRESS frames mentioning the same Address ID. This help the receiver figure out that a REMOVE_ADDRESS might have been sent before an ADD_ADDRESS frame implying the same Address ID, even if for some reason the REMOVE_ADDRESS reaches the receiver after the newer ADD_ADDRESS one.

A host SHOULD stop using sending uniflows using the removed address and set them in the UNUSED state. If the REMOVE_ADDRESS contains an Address ID that was not previously announced, the receiver MUST silently ignore the frame.

6.6. UNIFLOWS Frame

The UNIFLOWS frame (type=0x46) communicates the uniflows' state of the sending host to the peer. It allows the sender to communicate its active uniflows to the peer in order to detect potential connectivity issue over uniflows. It also enables hosts to map Address IDs to seen 4-tuples when middleboxes affecting them (e.g., NATs,...) are present.

The format of the UNIFLOWS frame is shown below.

```
UNIFLOWS Frame {
  Type (i) = 0x46,
  Sequence Number (i),
  Receiving Uniflows (i),
  Active Sending Uniflows (i),
  Receiving Uniflow Info Section (..) ...,
  Active Sending Uniflow Info Section (..) ...,
}
```

Figure 12: UNIFLOWS Frame Format

The UNIFLOWS frame contains the following fields.

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

Receiving Uniflows: The current number of receiving uniflows from the sender's point of view.

Active Sending Uniflows: The current number of sending uniflows in the ACTIVE state from the sender's point of view.

Receiving Uniflow Info Section: Contains information about the receiving uniflows (there are Receiving Uniflows entries).

Sending Uniflow Info Section: Contains information about the sending uniflows in ACTIVE state (there are Active Sending Uniflows entries).

Both Receiving Uniflow Info and Active Sending Uniflow Info Sections share the same format which is shown below.

```
Uniflow Info Section {
  Uniflow ID (i),
  Local Address ID (8),
}
```

Figure 13: Uniflow Info Section Format

The fields in the Uniflow Info Section are the following.

Uniflow ID: The Uniflow ID of the uniflow the sending host provides information about.

LocAddrID: The local Address ID of the address currently used by the uniflow.

The Uniflow Info section only contains the local Address ID so far, but this section can be extended later with other potentially useful information.

7. Extension of the Meaning of Existing QUIC Frames

The multipath extensions do not modify the wire format of existing QUIC frames. However, they extend the meaning of a few of them while keeping this addition transparent and consistent with the single-path QUIC design. The concerned frames (and their extended meaning) are the following.

NEW_CONNECTION_ID: Equivalent to a MP_NEW_CONNECTION_ID frame with Uniflow ID set to 0.

RETIRE_CONNECTION_ID: Equivalent to a MP_RETIRE_CONNECTION_ID frame with Uniflow ID set to 0.

ACK: Equivalent to a MP_ACK frame with Uniflow ID set to 0.

8. Security Considerations

8.1. Nonce Computation

With Multipath QUIC, each uniflow 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 uniflows on the same direction leads to the same cryptographic nonce. Using twice the same nonce MUST NOT happen, hence MP-QUIC has a different nonce computation than [I-D.ietf-quic-tls]

The left most bits of nonce MUST be the Uniflow ID that identifies the current uniflow up to max_sending_uniflow_id. The remaining bits of the nonce is formed by an exclusive OR of the least significant bits of the packet protection IV with the padded packet number (left-padded with 0s). The nonce MUST be left-padded with a 0 if max_sending_uniflow_id <= 2, and the max_sending_uniflow_id MUST NOT be higher than 2^61. If a uniflow has sent 2^62-max_sending_uniflow_id packets, another uniflow MUST be used to avoid re-using the same 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 uniflows to these addresses as described in Section 8 of [I-D.ietf-quic-transport]. Section 21.12.3 of [I-D.ietf-quic-transport] provides additional motivation for this process. In addition, hosts MUST send ADD ADDRESS frames in 1-RTT frames to prevent off-path attacks.

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 2 should be added to the "QUIC Transport Parameters" registry under the "QUIC Protocol" heading.

Value	Parameter Name	Specification
0x40	max_sending_uniflow_id	Section 5.1.1

Table 2: Addition to QUIC Transport Parameters Entries

10. Acknowledgments

We would like to thank Masahiro Kozuka and Kazuho Oku for their numerous comments on the first version of this draft. We also thank Philipp Tiesel for his early comments that led to the current design, and Ian Swett for later feedback. 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 and Florentin Rochet helped us to improve the last versions of this draft. This project was partially supported by the MQUIC project funded by the Walloon Government.

11. References

11.1. Normative References

[I-D.ietf-quic-invariants]
Thomson, M., "Version-Independent Properties of QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-invariants-13, 14 January 2021, <https://www.ietf.org/archive/id/draft-ietf-quic-invariants-13.txt>.

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", Work in Progress, Internet-Draft, draft-ietf-quic-recovery-34, 14 January 2021, <https://www.ietf.org/archive/id/draft-ietf-quic-recovery-34.txt>.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using TLS to Secure QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-tls-34, 14 January 2021, <https://www.ietf.org/archive/id/draft-ietf-quic-tls-34.txt>.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-34, 14 January 2021, <https://www.ietf.org/archive/id/draft-ietf-quic-transport-34.txt>.

[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>.

11.2. Informative References

[I-D.bonaventure-iccrg-schedulers]
Bonaventure, O., Piraux, M., Coninck, Q. D., Baerts, M., Paasch, C., and M. Amend, "Multipath schedulers", Work in Progress, Internet-Draft, draft-bonaventure-iccrg-schedulers-01, 9 September 2020, <https://www.ietf.org/archive/id/draft-bonaventure-iccrg-schedulers-01.txt>.

[I-D.huitema-quic-mpath-req]
Huitema, C., "QUIC Multipath Requirements", Work in Progress, Internet-Draft, draft-huitema-quic-mpath-req-01, 7 January 2018, <https://www.ietf.org/archive/id/draft-huitema-quic-mpath-req-01.txt>.

[IETFJ] Bonaventure, O. and S. Seo, "Multipath TCP Deployments", IETF Journal , November 2016.

[MFQUIC] De Coninck, Q. and O. Bonaventure, "Multiflow QUIC: A Generic Multipath Transport Protocol", To appear in IEEE Communications Magazine. A preprint is available at https://hdl.handle.net/2078.1/243486 , May 2021.

[MPQUIC] De Coninck, Q. and O. Bonaventure, "Multipath QUIC: Design and Evaluation", 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2017). http://multipath-quic.org , December 2017.

[MPRTP] Singh, V., Ahsan, S., and J. Ott, "MPRTP: Multipath considerations for real-time media", Proceedings of the 4th ACM Multimedia Systems Conference , 2013.

[OLIA] Khalili, R., Gast, N., Popovic, M., Upadhyay, U., and J.-Y. Le Boudec, "MPTCP is not pareto-optimal: performance issues and a possible solution", Proceedings of the 8th international conference on Emerging networking experiments and technologies, ACM , 2012.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC3077] Duros, E., Dabbous, W., Izumiyama, H., Fujii, N., and Y. Zhang, "A Link-Layer Tunneling Mechanism for Unidirectional Links", RFC 3077, DOI 10.17487/RFC3077, March 2001, <https://www.rfc-editor.org/info/rfc3077>.

[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>.

[RFC8041] Bonaventure, O., Paasch, C., and G. Detal, "Use Cases and Operational Experience with Multipath TCP", RFC 8041, DOI 10.17487/RFC8041, January 2017, <https://www.rfc-editor.org/info/rfc8041>.

[RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/info/rfc8684>.

[SCTPCMT] Iyengar, J., Amer, P., and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths", IEEE/ACM Transactions on networking, Vol. 14, no 5 , 2006.

Appendix A. Comparison with Multipath TCP

Multipath TCP [RFC8684] is currently the most widely deployed multipath transport protocol on the Internet. While its design impacted the initial versions of the Multipath extensions for the QUIC protocol, there are now major differences between both protocols that we now highlight.

A.1. Multipath TCP Bidirectional Paths vs. QUIC Uniflows

TCP ensures reliable data delivery by sending back acknowledgments to the data sender. Because data flows in one direction and acknowledgments in the other, the notion of bidirectional paths became a de facto standard with TCP. The Multipath TCP extension [RFC8684] combines several TCP connections to spread a single data stream over them. Hence, all the paths of a Multipath TCP connection must be bidirectional. However, networking experiences showed that packets following a direction do not always share the exact same road as the packets in the opposite direction. Furthermore, QUIC does not require a network path to be bidirectional in order to be used, as many parts of its design handle possible network asymmetries (unidirectional Connection IDs, PATH_RESPONSE that can flow on a different network path than the elliciting PATH_CHALLENGE,...).

A.2. Negotiating the Multipath Extensions

A Multipath TCP connection relies on the TCP option field of the TCP header to negotiate the multipath extensions. During the handshake, Multipath TCP uses the MP_CAPABLE TCP option to exchange connection's keys used to authenticate additional Multipath TCP subflows and generate tokens used to identify the connection. However, TCP options are sent in clear-text. Any on-path network observer may record the connection's keys and create Multipath TCP subflows on it. Multipath QUIC does not face this security issue. The multipath extensions are negotiated using authenticated transport parameters of the QUIC handshake. Then, Multipath QUIC leverages the encryption feature of QUIC to hide information from network observers.

A.3. Uniflow Establishment

To create additional subflows to a Multipath TCP connection, hosts initiate a TCP handshake by negotitating the MP_JOIN TCP option. This option carries a token matching the TCP subflow to a Multipath TCP connection and a HMAC value computed using the exchanged connection's keys. In addition that connection's keys were exchanged in clear-text during the handshake and that the token is also in clear-text, the HMAC value (using SHA-256) is truncated to the leftmost 64 bits (in SYN/ACK) or 160 bits (in third ACK) because of the TCP option length limitation to 40 bytes. Multipath QUIC avoids all these issues. The negotiation and usage of additional uniflows are performed using encrypted messages and the length of frames are only limited by the size of the packet that carries them.

Another difference comes in the control of the number of paths/ uniflows in use. Multipath TCP has a new subflow when the corresponding TCP handshake succeeds. However, it is not possible to restrict in advance the number of paths that a Multipath TCP connection can use. Therefore, the only way for a server to control the number of paths is to adopt a reactive approach, i.e., to reset or blackhole exceeding subflows from the client. In Multipath QUIC, each host sets a upper limit on the number of sending uniflows that it wants to use, while keeping control on the number of sending uniflows it provides to its peer. This path management is hence proactive.

A.4. Exchanging Data over Multiple Uniflows

One of the key design decision of Multipath TCP is that all its subflows have to behave like regular TCP connections to handle network interference. Each Multipath TCP subflow has to keep in-sequence data delivery and dedicates its TCP sequence number to this end. To handle multipath reordering, an additional Data Sequence Number at the Multipath TCP level is needed. In Multipath QUIC, the uniflow on which data is sent is a packet-level information. This means that a frame can be sent regardless of the uniflow of the packet carrying it. Furthermore, because the (STREAM) data offset is a frame-level information, there is no need to define additional sequence numbers to cope with reordering across uniflows.

In addition to this signaling overhead, Multipath TCP faces performance issues due to this acknoweldgment constraint. Consider the following scenario. Some data was first transmitted on a lossy path A, such that the peer never receives it. The sender can (successfully) retransmit the same data over a working path B (and gets the corresponding acknowledgment). However, the sender cannot send new data on the path A as long as the initial lost data was not delivered on that path (because of the TCP behavior constraint). Such transmission overhead is not present in Multipath QUIC, as there is no rule that a Multipath QUIC uniflow has to behave like a single-path QUIC one.

Another consequence of the "Multipath TCP subflows must behave like regular TCP connections" is that acknowkedgments have to stay on the same network path as the one used by the data. This constraint on the acknowledgment strategy is not present (and hardly enforceable) in Multipath QUIC as frames are independent of packets. Multipath QUIC can better benefit from high-latency paths and enable the usage of unidirectional networks.

A.5. Advertising Host's Addresses

Multipath TCP enables host to communicate their local IP addresses to its peer by using the ADD_ADDR TCP option. Similarly, REMOVE_ADDR TCP option is used to advertise the loss of a local address to its peer. Their usage is however subject to security issues, as these options are communicated in clear-text, possibly leaking the host's IP addresses to the network. This security concern does not affect Multipath QUIC as all this information is encrypted in frames.

Another point is related to the reliability of the address advertisement. In Multipath TCP, the ADD_ADDR and REMOVE_ADDR options are sent unreliably, i.e., there is no acknowledgment mechanism for their reception. While Multipath TCP [RFC8684]

provides an "echo" mechanism to the ADD_ADDR, there is no such equivalent for REMOVE_ADDR. In Multipath QUIC, ADD_ADDRESS and REMOVE_ADDRESS frames are ack-elliciting, making them reliable.

A.6. Congestion Control

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].

Appendix B. Change Log

B.1. Since draft-deconinck-quic-multipath-06

* Link with recent publication

B.2. Since draft-deconinck-quic-multipath-05

* Summarize frame types in a table
* Update frame format to structure style
* Update text to match draft-ietf-quic-transport-32
* Link to scheduling companion draft
* Remove dangling to-dos

B.3. Since draft-deconinck-quic-multipath-04

* Mostly editorial and reference fixes

B.4. Since draft-deconinck-quic-multipath-03

* Clarify the notion of asymmetric paths by introducing uniflows
* Remove the PATH_UPDATE frame
* Rename PATHS frame to UNIFLOWS frame and adapt its content
* Add a sequence number to frames involving Address ID events (#4)
* Disallow Zero-length connection ID (#2)
* Correctly handle nonce computation (thanks to Florentin Rochet)

* Focus on the core concepts of multipath and delegate algorithms to companion drafts

* Updated text to match draft-ietf-quic-transport-27

B.5. Since draft-deconinck-quic-multipath-02

* Consider asymmetric paths

B.6. 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

B.7. 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

B.8. 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