QUIC Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

Q. De Coninck
O. Bonaventure
UCLouvain

March 5, 2018

Multipath Extension for QUIC
draft-deconinck-quic-multipath-00

Abstract

This document proposes extensions to the QUIC protocol to enable the usage of multiple paths over a single connection. Those changes 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

Copyright Notice

Table of Contents

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 bandwidth. During the last years, several multipath extensions to transport protocols have been proposed [RFC6824],[MPRTP],[SCTPCMT]. Multipath TCP [RFC6824] is the most mature one. It is already deployed on popular smartphones and for other use cases [RFC8041].

With regular TCP and UDP, all the packets that belong to a given flow contain the same 5-tuple that acts as an identifier for this flow. This prevents such 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 its Connection ID. 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 use multiple paths. Use cases such as bandwidth aggregation or seamless network handovers would be applicable to QUIC, as they are now with Multipath TCP. A first performance evaluation of such use cases and a comparison between Multipath QUIC and Multipath TCP may be found in [MPQUIC].

In this draft, we leverage many of the lessons learned from the design of Multipath TCP and comments received on the first version of this draft to propose extensions to the current QUIC design to enable it to simultaneously use several paths. 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 the QUIC documents [I-D.ietf-quic-transport]. In addition, we define:

- o Path: A logical association between two hosts over which packets can be sent. A path is internally identified by a Path ID and

uses a potentially changing Connection ID in packets exchanged on that path.

o Initial Path: The path used for the establishment of the QUIC connection. The cryptographic handshake is done on this path. It is 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], with the following additional convention:

x (j) ... Indicates that x uses the variable-length encoding scheme described in Section 5.2.

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 WiFi 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. 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". A path is a UDP flow between two hosts denoted by a 4-tuple (source IP address, destination IP address, source port, destination port). On a smartphone interacting with a single-homed server, the mobile device might want to use one path over the WiFi 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 WiFi network, one over IPv4 and the other over IPv6.

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. Another of its use cases is fail-over when the address in use fails but another one is available. A mobile device loosing the WiFi connectivity can then continue the connection over its cellular interface.

A QUIC connection can thus start on a given path and end on another one. However, the current QUIC design [I-D.ietf-quic-transport] assumes that only one path is in use for a given connection. Instead of switching the 4-tuple for the whole connection, this draft first proposes mechanisms to communicate endhost addresses to the peer. It then leverage 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. 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.

```
   Server                          Phone                          Server
   via WiFi                                                      via LTE
   -------                        -------                          -----
     | Pkt(CID=A,PN=1,frames=[       |                               |
     |  STREAM("Request (1/2)")])    | Pkt(CID=B,PN=1,frames=[       |
     |<------------------------------|  STREAM("Request (2/2)")])    |
     | Pkt(CID=A,PN=1,frames=[       |--------                       |
     |  ACK(PID=1,LargestAcked=1)])  |       |----------             |
     |------------------------------>|                 |----------   |
     | Pkt(CID=A,PN=2,frames=[       |                           |-->|
     |  STREAM("Response 1")])       | Pkt(CID=B,PN=1,frames=[       |
     |------------------------------>|  ACK(PID=2,LargestAcked=1),   |
     |                               |  STREAM("Response 2")])   ----|
     | Pkt(CID=A,PN=2,frames=[       |                 ----------|   |
     |  ACK(PID=1, LargestAcked=2),  |       ----------|             |
     |  ACK(PID=2, LargestAcked=1)]) |<------|                       |
     |<------------------------------|                               |
     | Pkt(CID=A,PN=3,frames=[       | Pkt(CID=B,PN=2,frames=[       |
     |  STREAM("Response 3")])       |  STREAM("Response 4")])       |
     |------------------------------>|                           ----|
     |                               |                 ----------|   |
     |            ...                |  ...  <---------|             |
```

Figure 1: Dataflow with Multipath QUIC

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 over Stream ID 0 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_path_id transportparameter, where both hosts advertise the maximum number of paths, in addition to the first one, that they are able to use over the connection. The negotiated value will determine how many bytes will be used to encode the Path ID in frames. If one of the hosts does not advertise the max_path_id transport parameter, the negotiated value is 0, meaning that the QUIC connection will remain single-path.

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 one or more paths. Each path is associated with a different four-tuple and identified by a Path ID, as shown in Figure 2.

```
          +-----------------------------------------------+
          |               Connection (MCID)               |
          |   +--------+ +--------+ ... +------------+   |
          |   | Path 0 | | Path 1 |     | Path N - 1 |   |
          |   | Tuple  | | Tuple' | ... |   Tuple"   |   |
          |   |  PCID  | |  PCID' |     |    PCID"   |   |
          |   |  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 draft proposes two levels of Connection IDs. The first one is the Master Connection ID (MCID). It uniquely identifies the connection, as with the current QUIC design. The second one is the Path Connection ID (PCID). Packets belonging to a given path share the same PCID written in the Connection ID field of the public header. The PCID acts as an implicit path identifier for packets. Preventing the linkability of different paths is an important requirement for the multipath extension [I-D.huitema-quic-mpath-req]. Using PCID as implicit path identifier makes this linkability harder than having explicit signaling as in the previous version of this draft and does not require public header change to keep invariants [I-D.ietf-quic-invariants]. The MCID of a connection will be the PCID of the Initial Path. In the example of Figure 1, if the connection started on the WiFi network, then Connection ID A is both the PCID of the WiFi path and the MCID of the connection.

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

3.5. Path Establishment

The max_path_id transport parameter exchanged during the cryptographic handshake determines an upper bound on the number of paths that can be created over the connection, in addition to the

Initial one. Then, hosts must agree on which paths can be used, and with which Path Connection IDs. Unlike Multipath TCP [RFC6824], the server dynamically controls how many paths can currently be in use. This can be done using a new MAX_PATHS frame indicating how many additional paths can be simultaneously in use.

The server proposes new paths with an extended version of the NEW_CONNECTION_ID frame (see Section 6.1). That frame proposes a PCID for a given Path ID. Once the client receives and acknowledges the frame, it can start using the new path by placing the PCID in the Connection ID field of the packets. The server can start using a path once the corresponding NEW_CONNECTION_ID has been acknowledged. Both hosts store the NEW_CONNECTION_ID information in order to cope with the remote trying to use a new path. As frames are encrypted, the establishment of new paths does not leak cleartext identifiers [I-D.huitema-quic-mpath-req].

A server might provide more Path IDs with NEW_CONNECTION_ID frames than the value advertised in the MAX_PATHS frame. This can be useful to cope with migration cases, as described in Section 3.9. In such cases, hosts can only use a subset of the proposed Path IDs. Multipath QUIC is fully symmetrical. Both the client and the server can start using new paths once their corresponding PCID have been negotiated. It might happen that both hosts try to create paths with different Path IDs, such that there are more paths than the advertised number in MAX_PATHS frame. In that case, the paths with the lowest Path IDs will be used while the others will stop.

The client is not able to create new paths as long as the server does not send NEW_CONNECTION_ID and MAX_PATHS frames. To limit the latency of the path handshake, the server can send those frames just after the 0-RTT handshake packet.

Although a path can be first used by any host, it might not be practical for one of the peers to send an initial packet on new paths. A possible cause 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 path using it.

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 the Path ID in the public header 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 2^32 * 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 path they want to send path acknowledgments. Looking again at Figure 1, packets that were sent over a given path (e.g., the response2 packet on path 2 with CID B) can be acknowledged on another path (here, path 1 with CID A) to limit the latency due to ACK transmissions on high-latency paths. Such scheduling decision would not have been possible in Multipath TCP [RFC6824] which must acknowledge data on the path is was received on.

3.7. Exchanging Addresses

When a multi-homed mobile device connects to a dual-stacked server on its IPv4 address, it is aware of its local addresses (e.g., the WiFi 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 address to start a new 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 of the sending host. A path is called active when it was created over a validated 4-tuple and is still in use. The frame indicates the local Address ID that the path uses. With this information, the server can then validate the public address and associate the advertised 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 reachability of a WiFi 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 connection gathers a set of paths, each using 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 path. There remain situations where this change is unavoidable. These can be categorized into two groups: host-aware changes (e.g., network handover from WiFi 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 WiFi and cellular. It advertised both addresses and the server currently enables only one path, the Initial one. The Initial Path uses the WiFi address. Then, for some reason, the WiFi 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 WiFi 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 the 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 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 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 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.

3.10. Congestion Control

The QUIC congestion control scheme is defined in [I-D.ietf-quic-recovery]. This congestion control scheme is not suitable when several paths are active. Using the congestion control scheme defined in [I-D.ietf-quic-recovery] with Multipath QUIC would result in unfairness. Each 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 ID

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

The Master Connection ID is determined during the cryptographic handshake and actually corresponds to the Connection ID in the current QUIC design [I-D.ietf-quic-transport]. The Path Connection ID of the Initial path (with Path ID 0) is equal to the Master Connection ID. The Path Connection ID 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 the PCID, the server can start receiving packets with the advertised Connection ID as belonging to the corresponding path. The server MUST wait until the reception of the frame acknowledgment before starting to send packets on that path.

Upon reception of the NEW_CONNECTION_ID frame, the client MUST acknowledge it and MUST store the advertised Path Connection ID and the Path ID of the proposed path. It MUST ensure that it can receive packets coming from the server using the PCID and associate it with the corresponding Path ID.

5. Using Multiple Paths

This section describes in details the multipath operations with the QUIC protocol.

5.1. Multipath Negotiation

The Multipath Negotiation takes place during the cryptographic handshake with the max_path_id 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 the maximum path ID they are willing to use. The maximum path ID that can be used over a connection is the minimum of both advertised values. If one host does not send the max_path_id transport parameter during the cryptographic handshake, the remote MUST assume a value of 0, leading to a single-path connection over the Initial Path. NEW_CONNECTION_ID frames can later propose Path IDs in the range of max_path_id inclusive. If a client receives a NEW_CONNECTION_ID proposing a Path ID greater than the negotiated maximum value, it MUST trigger a CONNECTION_CLOSE frame with an error code set to 0x10b with a reason phrase describing this issue.

5.1.1. Transport Parameter Definition

An endhost MAY use the following transport parameter:

max_path_id (0x0020): The maximum path ID transport parameter indicates the number of paths in addition to the Initial Path that the host agrees to support over the connection, encoded as an unsigned 32-bit integer. This sets a upper bound on the Path IDs that can be used. The default value for this parameter is 0, meaning that a host omitting this transport parameter does not agree to use multiple paths over the connection.

5.2. Variable-Length Encoding of the Path ID

From the negotiated maximum Path ID, hosts can infer the amount of bits required to encode the Path ID. If the maximum Path ID is 255, only one byte suffices. Similarly, if one of the hosts wants a single-path connection, either by advertising a max_path_id of 0 or by omitting it, the Path ID field should not be present.

Negotiated Max Path ID Range	Path ID Field Length (Bytes)
0	0
1-255	1
256-65535	2
65536-4294967295	4

Table 1: Summary of Path ID Encodings

The proposed encoding is summarized in Table 1.

Because of security considerations described in Section 8.1, a host reusing packet numbers across paths must limit the number of advertised paths depending on the size of the Initial Value. In particular, a host MUST NOT advertise a max_path_id whose field length exceeds (Initial Value Length) - 8.

5.3. Coping with Additional Remote Addresses

The usage of multiple networks paths if often done using multiple IP addresses. For instance, a smartphone willing to use both WiFi and LTE will use the corresponding addresses provided by the networks. It can then safely send packets to a previously-used IP address of a

server. The server can receive packets with different IPs, 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.

5.4. Path State

During the Multipath QUIC connection, hosts maintain some state for paths. Information about the path that hosts are required to store depends on its state. The possible path states are depicted in Figure 3.

```
   +----------+
   | PROPOSED |
   +----------+
         |
         | path usage
         |
         |       restricted by MAX_PATHS
         v    or affected by REMOVE_ADDRESS
    +--------+ --------------------------> +----------+
    | ACTIVE |       reusing path          | UNUSABLE |
    +--------+ <-------------------------- +----------+
         |                                       |
         | PATH_UPDATE                           |
         +---------------------------------------+
         |
         v
    +--------+
    | CLOSED |
    +--------+
```

Figure 3: Finite-State Machine of paths

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

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

- Master Connection ID: encoded as a 8-byte number. It makes the link between the path and the QUIC connection it belongs to. This value is immutable.

- Path Connection ID: encoded as a 8-byte integer. It makes the link between the packet's Connection ID field and the path. This value can be updated with subsequent NEW_CONNECTION_ID frames.

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

Hosts MUST ensure that they can associate a packet with the PCID and its corresponding path at any time, as they must be ready to figure out the path used by the remote host.

Either the host received a packet with the corresponding PCID coming from a validated address or it wants to start using the path to a validated address, the path goes to the ACTIVE state. This is the state where a path can be used to send and receive packets. In addition to the fields required in the PROPOSED state, the following elements MUST be tracked:

- Packet Number Space: each path is associated with its own monotonically increasing packet number space. Each endpoint maintains a separate packet number for sending and receiving packets. Packet number considerations described in [I-D.ietf-quic-transport] apply within a given path.

- Current 4-tuple: the tuple (sIP, dIP, sport, dport) used to send or receive
packets over this path. This value is mutable, either because the host decides to change its local address and/or port, or because it receives a packet with a different validated remote address and/or port than the one currently recorded. A host that changes the 4-tuple of a 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 mark a path as UNUSABLE 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 round-trip-time or number of packets sent and received 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 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 is in UNUSABLE state and the host MUST NOT send packets on it. The host keeps the same information as in the ACTIVE state for the path. It might happen that packets are received on that path. If the path is in UNUSABLE state because of a decrease of the active paths, packets MUST be discarded silently. If it is because a REMOVE_ADDRESS was received for the remote Address ID of the path, the host SHOULD validate the remote address first before reusing it. If this validation suceeds, or the number of paths increased again, a path in the UNUSABLE state can go into ACTIVE state, but its congestion window MUST be restarted and its performance metrics SHOULD be reset.

Eventually, a path may either be migrated to another one or closed. This is signalled by the PATH_UPDATE frame. In that case, the path is in CLOSED state. In that state, packets MUST NOT be sent or received over it. A host MUST keep the same path state field as in the PROPOSED state, to avoid ambiguities and CLOSED path reuse.

5.5. Dynamic Management of Paths

The server is the main protagonist in determining how many paths can be used over a Multipath QUIC connection. It is its reponisbility to propose Path IDs with corresponding PCIDs that could be used by the client. In addition, it dynamically controls the number of active paths that can be used for the connection. This is performed by sending the MAX_PATHS frame that sets an upper limit on the number of additional active paths. A host can propose more Path IDs with NEW_CONNECTION_ID frames than the number of additional paths it agrees to use simultaneously. This can be useful in migration scenarios, where the server wants to share a pool of Path IDs that can be directly used to migrate paths.

As long as no MAX_PATHS frame was received from the server, the client MUST assume that only one path can be used. At the beginning of the QUIC connection, this means that only the Initial Path can be used. In this case however, during the connection, another path can be used, but the Initial Path MUST have been migrated to it. The server can then increase the value advertised in the MAX_PATHS frame to allow hosts to use new paths among the proposed Path ID pool.

By default, the client does not set any limit to the number of paths which can be used. If a client wants to limit the number of (additional) paths the server can use, it can also send a MAX_PATHS frame. In this case, the number of additional paths that can be used over the connection is the minimum between last advertised value from both sides.

A host can also decrease the number of paths that can be used over a connection. If this value decreases below the current number of active paths, hosts MUST put in UNUSABLE state the paths in the ACTIVE state with the highest Path IDs. Server and client MUST stop sending packets over UNUSABLE paths once the MAX_PATHS has been sent or received. The host restricting the number of paths SHOULD allow receiving packets on newly UNUSABLE paths for one round-trip-time. After this delay, hosts SHOULD silently discard received packets.

When the server proposes more Path IDs than the negotiated maximum number of ACTIVE paths, it might happen that both hosts decide at the same time to create paths with different Path IDs, such as there are more created paths than the negotiated maximum value. In this case, the hosts MUST only keep as ACTIVE the paths with the lowest Path IDs. Paths with the highest Path IDs are in the UNUSABLE state and packets SHOULD be silently discarded.

5.6. Loosing Addresses

During the lifetime of a connection, a host might lose addresses, e.g., a network interface that was shut down. All the ACTIVE paths that were using that local address MUST be set in the UNUSABLE state. 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 set the ACTIVE paths affected by the address removal into the UNUSABLE state.

The host that locally lost the address MAY reuse one of these 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 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 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 path on which ACK frames are sent impacts the peer. The ACK frame is notably used to determine the latency of a path. If the ACK frame is sent on the path it acknowledges, then the peer can compute the round-trip-time of that path. Otherwise, the peer would compute the latency as the sum of the foward delay of the acknowledged path and the return delay of the path used to send the ACK frame. Choosing between acknowledging packets on the same path or on a specific path 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. 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 is modified to provide Path Connection ID negotiation for each path. The ACK frame contains packet-level information with the Largest Acknowledged field. Since the Packet Numbers are now associated to 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 server to indicate which connection IDs the client must use to take advantage of multiple paths.

The NEW_CONNECTION_ID is as follows:

```
    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 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Path ID  (j)                      ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Sequence (i)                      ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                        Connection ID (64)                     +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                                                               |
   +                   Stateless Reset Token (128)                 +
   |                                                               |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: NEW_CONNECTION_ID adapted to Multipath QUIC

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 MUST only be sent by the server. Upon reception of the frame with a specified path ID, the client can start using the new path with the matching Connection ID.

A host MUST NOT start using a path if no related NEW_CONNECTION_ID frame has been received (at client side) or acknowledged (at server side). To limit the delay of the multipath usage upon handshake completion, the server SHOULD send NEW_CONNECTION_ID frames for paths it allows 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 the server MUST be random.

It might happen that a server proposes a Connection ID in a NEW_CONNECTION_ID frame that is already in use by the client for another connection with another server. In such cases, the client MUST warn the server that this Connection ID is not available and ask for another Connection ID. This request for asking another

Connection ID for a particular path will follow the same mechanism as described in [I-D.ietf-quic-transport].

6.2. 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Path ID (j)                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Largest Acknowledged (i)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          ACK Delay (i)                      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       ACK Block Count (i)                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          ACK Blocks (*)                     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: 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.

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. MAX_PATHS Frame

The MAX_PATHS frame is used by hosts to control the number of paths that can be simultaneously in the ACTIVE state on a given connection. The proposed type for the MAX_PATHS frame is 0x20. A MAX_PATHS 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 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Sequence (i)                      ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                 Num Additional Active Paths (j)             ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: MAX_PATHS Frame

The MAX_PATHS frame contains the following fields:

o Sequence: A variable-length integer. This value starts at 0 and increases by 1 for each change of number of active paths that is provided by the host.

o Num Additional Active Paths: A variable-length integer indicating how many additional paths can be used over the connection. The number of paths that can be in ACTIVE state is thus (Num Additional Active Paths + 1).

Once the connection is established, hosts MUST assume that the server sent a MAX_PATHS frame with sequence -1 and number of additional active paths equal to 0 and the client sent a MAX_PATHS frame with sequence -1 and number of additional active paths equal to the negotiated max_path_id value.

7.2. **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
    0 1 2 3 4 5 6 7 8 9 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 (j)                  ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Proposed Path ID (j)                 ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: MIGRATE_PATH Frame

The MIGRATE_PATH frame contains the following fields:

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

- o 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 have been either in the PROPOSED or in the UNUSABLE state and MUST now be considered as ACTIVE.

7.3. 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0|0|P|IPVers.|Address ID (8) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       IP Address (32/128)                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          [Port (16)]          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: ADD_ADDRESS Frame

The ADD_ADDRESS frame contains the following fields:

- o Reserved bits: the three most-significant bits of the first byte are set to 0, and are reserved for future use.

- o P bit: the fourth most-significant bit of the first byte indicates, if set, the presence of the Port field.

- o IPVers.: the remaining four least-significant bits of the first byte contain the version of the IP address contained in the IP Address field.

- o 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.

- o IP Address: the advertised IP address, in network order.

- o 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.

7.4. 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|Address ID (8) |
+-+-+-+-+-+-+-+-+
```

Figure 9: REMOVE_ADDRESS Frame

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.

7.5. 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.

```
     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 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          Sequence (i)                       ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        ActivePaths (j)                      ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Path Info Section (*)                   ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: PATHS Frame

The PATHS frame contains the following fields:

o 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.

o ActivePaths: the current number of additional paths considered as being active from the sender point of view, i.e., (the number of active paths - 1). ActivePaths MUST be lower or equal to the last value advertised by MAX_PATHS frame.

o Path Info Section: contains information about all the active paths (i.e., there are ActivePaths + 1 entries). The format of this section 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 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         Path ID 0 (j)                       ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |LocAddrID 0 (8)|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         Path ID N (j)                       ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |LocAddrID N (8)|
    +-+-+-+-+-+-+-+-+
```

Figure 11: Path Info Section

The fields in the Path Info Section are:

o Path ID: the Path ID of the active path the sending host provides information about.

- o LocAddrID: the local Address ID of the address currently used by the path.

The Path Info section only contains the Local Address ID 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 described in Section 5.2 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 2 should be added to the "QUIC Transport Parameters" registry under the "QUIC Protocol" heading.

Value	Parameter Name	Specification
0x0020	max_path_id	Section 5.1.1

Table 2: 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. We also want to thanks Christian Huitema for his draft about multipath requirements to identify critical elements about the multipath feature.

11. References

11.1. Normative References

[I-D.ietf-quic-invariants]
Thomson, M., "Version-Independent Properties of QUIC", draft-ietf-quic-invariants-00 (work in progress), February 2018.

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-09 (work in progress), January 2018.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-10 (work in progress), March 2018.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-10 (work in progress), March 2018.

[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>.

11.2. Informative References

[Cellnet] Paasch, C., Detal, G., Duchene, F., Raiciu, C., and O. Bonaventure, "Exploring Mobile/WiFi Handover with Multipath TCP", ACM SIGCOMM workshop on Cellular Networks (Cellnet'12) , 2012.

[I-D.huitema-quic-mpath-req]
Huitema, C., "QUIC Multipath Requirements", draft-huitema-quic-mpath-req-01 (work in progress), January 2018.

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

[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. 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>.

[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>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <https://www.rfc-editor.org/info/rfc6824>.

[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>.

[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. Change Log

A.1. Since draft-deconinck-multipath-quic-00

o Added PATH_UPDATE frame

o Added MAX_PATHS frame

o No more packet header change

o Implicit Path ID notification using Connection ID and NEW_CONNECTION_ID frames

o Variable-length encoding for Path ID

o Updated text to match draft-ietf-quic-transport-10

o Fixed various typos

Authors' Addresses

Quentin De Coninck
UCLouvain

Email: quentin.deconinck@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: Olivier.Bonaventure@uclouvain.be