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

Abstract

   Multipath TCP has shown how a reliable transport protocol can
   efficiently use multiple paths for a given connection. We leverage
   the experience gained with Multipath TCP to propose simple extensions
   that enable QUIC to efficiently use multiple paths during the
   lifetime of a QUIC connection.

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 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 flows to survive to events such as NAT rebinding or
mobility cases where the IP address or the port of one of the
communicating peer changes. This connection migration feature is key
for QUIC to migrate a flow from one path to another. However, this
path change is implicit and the current design
[I-D.ietf-quic-transport] still assumes single-path flows. Seamless
handovers between wireless networks on smartphones are one of the
motivations for connection migration in QUIC. However, experience
with Multipath TCP shows that the handover between different wireless
networks is not an abrupt process [Cellnet],[IETFJ] . To support
seamless handovers, it is important to be able to use two (or more)
paths simultaneously during the handover.

Bringing Multipath to QUIC allows hosts to aggregate several networks
while providing better handover support, and potentially opens new
use cases. A detailed performance evaluation 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 propose extensions to the current QUIC
design to enable it to simultaneously use several paths. This
document is organized as follows. It first provides an overview of
the operation of Multipath QUIC. It then states changes required in
the packet format and specifies the usage of multiple paths. It also
defines new frames to perform multipath operations. Finally, it
provides some security and IANA 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 identified by a Path ID.

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.

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 them describe it at a high
   level.

3.1. What is a Path?

   Before going into details, let's first define what is called a
   "path". A path is a UDP flow between two hosts denoted by a 4-tuple
   (IP source address, IP destination address, source port, destination
   port). On a smartphone interacting with a single-homed server, the
   mobile device could decide to use one path over the WiFi network and
   another over the cellular one. Those paths are not necessarily
   completely 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 one over IPv6.

3.2. Going Further than 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 QUIC packets.
   This enables hosts to continue the connection even if the 4-tuple
   changed 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. This
   connection migration feature is not intended to support the
   simultaneous usage of multiple paths. To illustrate this point,
   consider the following scenario where a smartphone connected to both

   WiFi and LTE networks sends a POST request fitting in 2 packets and
   receives a large response from the single-homed server.

```
   Server                          Phone                         Server
   via WiFi                                                      via LTE
   -------                         -------                       -----
      |   Pkt(CID=A, PN=1, frames=[  |
      |     STREAM("Request (1/2)")]) | Pkt(CID=A, PN=2, frames=[      |
      |<---------------------------|   STREAM("Request (2/2)")])   |
      |   Pkt(CID=A, PN=1, frames=[  |--------                        |
      |    ACK(LargestAcked=1)])     |      |----------               |
      |--------------------------->|               |----------        |
      |                            |                         |-->|
      |                            |   Pkt(CID=A, PN=2, frames=[   |
      |                            |    ACK(LargestAcked=2),        |
      |                            |    STREAM("Response 1")]) ----|
      |                            |               ----------|       |
      |                            |       ----------|               |
      |                            |<------|                         |
      |                            |   Pkt(CID=A, PN=3, frames=[   |
      |                            |    ACK (LargestAcked=2)])     |
      |                            |--------                        |
      |                            |      |----------               |
      |                            |               |----------      |
      |                            |                         |-->|
      |                            |   Pkt(CID=A, PN=2, frames=[   |
      |                            |    STREAM("Response 2")])     |
      |                            |                         ----|
      |                            |        ...   <---------|   |
```
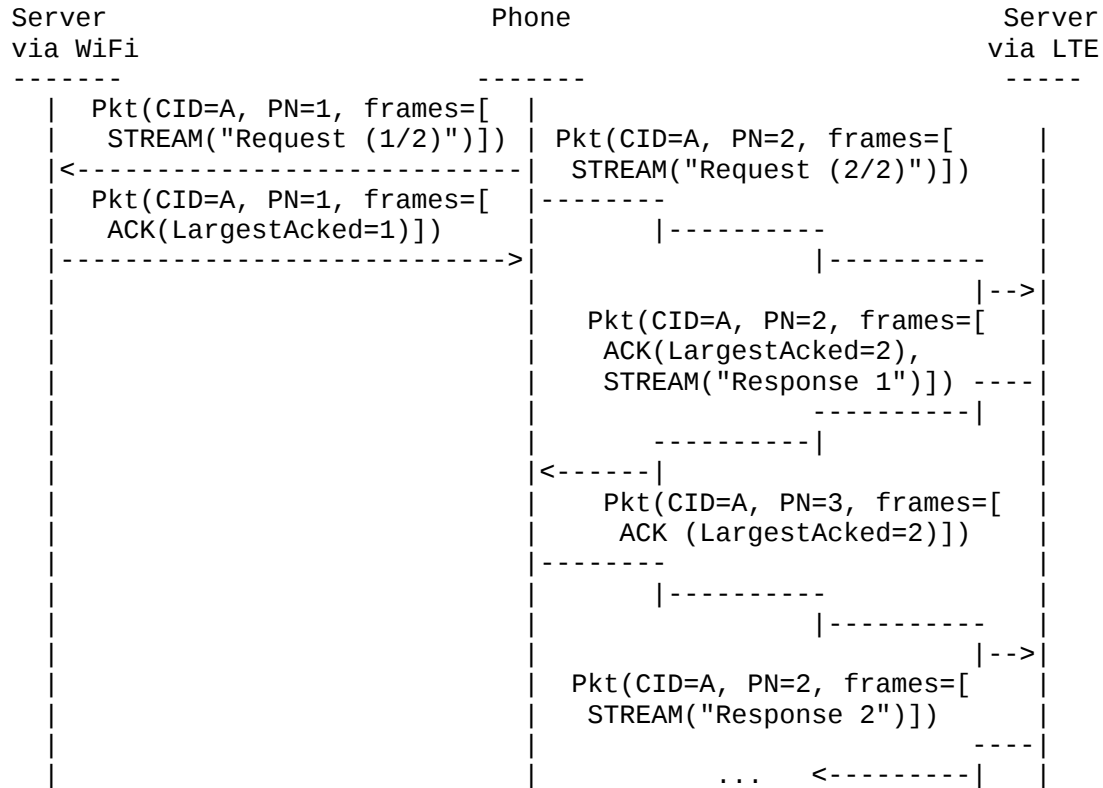
                 Figure 1: Single-path QUIC with multiple paths

   Assume the client wants to aggregate all the network interfaces it
   has. It sends the first packet with the STREAM frame containing the
   beginning of the request on the WiFi interface. It then sends the
   second one on its LTE interface. If the WiFi network exhibits a
   lower latency than the LTE one, the server will first receive the
   packet from the WiFi network and acknowledge it by sending an ACK
   frame on the WiFi. Then, it receives the second packet from the LTE
   network. Thanks to the Connection ID (CID) in the public header of
   the QUIC packet, the server detects that this packet belongs to the
   same connection and performs a connection migration over this new
   4-tuple. At this point, the connection remains stuck on the cellular
   network, unless the smartphone sends a new packet over the WiFi
   network. This is because there is currently no way for the server to
   know that the remote peer uses two different networks paths with
   potentially different properties. The smartphone might have sent the

first response ACK frame on the WiFi to drift the connection towards
the WiFi network, but the server would lose its ability to use the
cellular one, and would possibly observe large RTT variance over the
connection. The multipath extension of QUIC aims to achieve an
efficient usage of multiple paths by making them explicit to peers.

With the proposed multipath extension to QUIC, the example presented
in Figure 1 can become the one presented below.

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

Figure 2: Dataflow with Multipath QUIC

With the notion of multiple paths explicitly advertised, the server
is aware that multiple paths using different networks are present,
and can potentially be used simultaneously since it knows the 4-tuple
to use for each path. When the server receives the request that was
carried over two different paths, it can then use both of them to
transfer back the response to the client. The remaining of this
section focuses on giving a high-level view of the multipath
operations in QUIC.

3.3. Starting a Multipath QUIC Connection

Before using multiple paths, the QUIC connection must be established.
A Multipath QUIC connection always starts over an initial path where
the cryptographic handshake takes place over the dedicated stream

with Stream ID 0. The establishment is thus performed as in the
current QUIC design [I-D.ietf-quic-transport] [I-D.ietf-quic-tls].
The negotiation of multipath is performed during the cryptographic
handshake with the max_path_id transport parameter, where both hosts
advertise how many paths they are willing to use. The number of
paths that can then be used over the connection once handshake
completes is the minimum between both advertised values. The path on
which the cryptographic handshake and the path number negotiation are
performed is called the Initial Path and is identified by the Path ID
0.

3.4. Multipath QUIC Architecture

   Once the connection is established, QUIC can start to use as many
   paths as negotiated. A Multipath QUIC connection is composed of
   several paths. Each path is associated with a different four-tuple
   and identified by a Path ID, as shown in Figure 3.

```
        +-----------------------------------------------+
        |                  Connection                   |
        |   +--------+ +--------+ ... +------------+     |
        |   | Path 0 | | Path 1 |     | Path N - 1 |     |
        |   | Tuple  | | Tuple' | ... |   Tuple"   |     |
        |   |  PN    | |  PN'   |     |    PN"      |     |
        |   +--------+ +--------+ ... +------------+     |
        +-----------------------------------------------+
```

          Figure 3: Architectural view of Multipath QUIC

   A Multipath QUIC connection starts on the Initial Path, identified by
   Path ID 0. In Multipath QUIC, each packet (except the handshake
   packets at the beginning of the connection) explicitly contains the
   path identifier of the path it belongs to. This Path ID is included
   in the QUIC public header of each packet. This design simplifies the
   detection of packet losses on a per-path basis and enables a receiver
   to easily detect out-of-order packets on a given path. Hosts can
   also collect network information about each path, such as round-trip-
   time measurements and maintain a per-path congestion window.

3.5. Exchanging Data over Multiple Paths

   A QUIC packet acts as a container of 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.

However, 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 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 256
* 2^{64} packets on a QUIC connection since each path (with a Path ID
encoded on 1 byte) has its own packet number space.

The ACK frame is also modified to allow per-path packet
acknowledgments. This remains compliant with the design decision of
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 in Figure 2, packets that were sent
over a given path (e.g., the response2 packet on path 3) can be
acknowledged on another path (here, path 1) to limit the latency due
to ACK transmission on high-latency paths. Such scheduling decision
would not have been possible in Multipath TCP [RFC6824] which must
acknowledge data coming from a given path on the same path.

3.6. Starting to Use Paths

The cryptographic handshake determines how many paths, in addition to
the initial one, can be used. Once this handshake completes, hosts
can start using them simply by transmitting QUIC packets with the
associated Path ID. In contrast with Multipath TCP, Multipath QUIC
does not require a per-path handshake. This reduces the time
required to use a not used yet path. Multipath QUIC is fully
symmetrical. Both client and server can start using new paths.

Although a path can be first used by any host, it might not be
practical for one of the peers to start using new paths. A possible
cause is when a server wants to initiate the usage of a new path to a
NAT'd client. The client would possibly not receive the packet,
leading to connectivity issues on that path. To detect such issues,
the PATHS frame provides a list of the current active paths of the

   sending hosts to the peer. A path is called active when a functional
   network 4-tuple (on which either packets were sent or received on it
   or host received acknowledgments for packets over that path) is
   assigned to it. Furthermore, the PATHS frame indicates which 4-tuple
   the path is currently using. It also contains some path status
   metrics such as the round-trip-time estimated by the sending host
   over a given path in order to provide a global view of the path
   performance.

3.7. Communicating New 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
   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 has an Address ID given by the sending host. The
   addresses assigned to a host can vary during the lifetime of a
   Multipath QUIC connection. A new ADD_ADDRESS frame is transmitted
   when 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.

3.8. Path Migration

   At a given time, a Multipath QUIC connection gathers a set of paths,
   each denoted by a 4-tuple. The 4-tuple that is associated to a path
   is not fixed. It may change during the lifetime of a connection.
   Those changes can be caused by NAT rebindings or handovers for
   example.

```
                                                                    Host
                                                                    ----
   UDP(sIP= 192.0.2.0, dIP=198.51.100.0, sport=S, dport=D, payload=  |
        QUICPkt(CID=A, PN=X, PathID=1))                              |
             ----------------------------------------------------------> |
                                                                    |
   UDP(sIP= 198.51.100.0, dIP=192.0.2.0, sport=D, dport=S, payload=  |
        QUICPkt(CID=A, PN=Y, PathID=1))                              |
             <---------------------------------------------------------- |
                                                                    |
   UDP(sIP= 203.0.113.0, dIP=198.51.100.0, sport=T, dport=D, payload= |
        QUICPkt(CID=A, PN=X+1, PathID=1))                            |
             ----------------------------------------------------------> |
                                                                    |
   UDP(sIP= 198.51.100.0, dIP=203.0.113.0, sport=D, dport=T, payload= |
        QUICPkt(CID=A, PN=Y+1, PathID=1))                            |
             <---------------------------------------------------------- |
                                                                    |
```
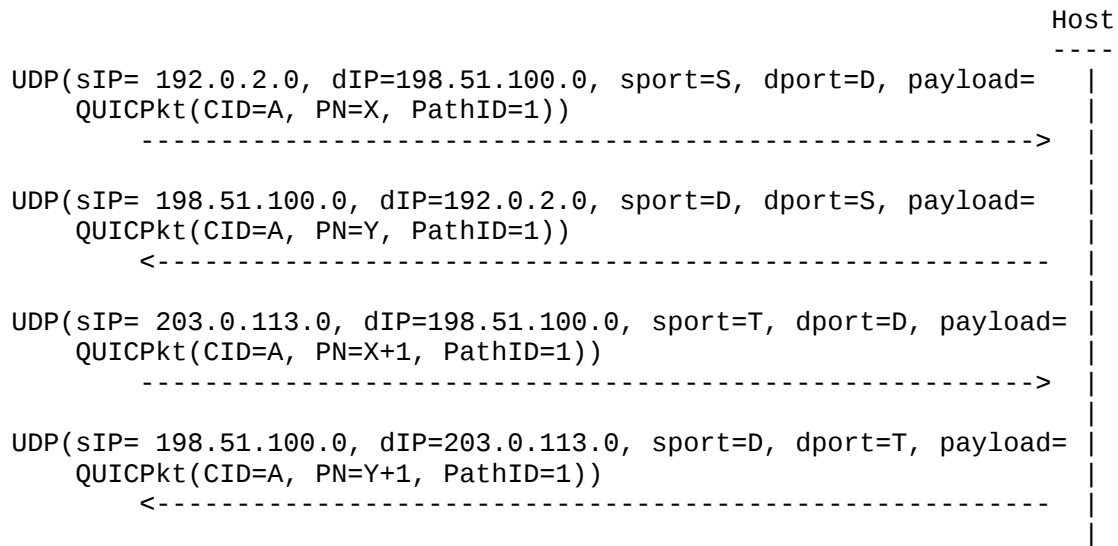
                    Figure 4: Example of path migration

   Figure 4 shows an example of path whose 4-tuple changes during the
   connection. Initially, from the host perspective, path 1 is using
   the 4-tuple (198.51.100.0, 192.0.2.0, D, S). The first packet it
   receives still uses it and the host uses that 4-tuple to generate the
   next packet on that path. Then, host receives a packet on path 1
   with a different 4-tuple (198.51.100.0, 203.0.113.0, D, T), with both
   changed remote IP and port. Because the QUIC packet contains both
   the Connection ID and the Path ID it belongs to, host can simply
   adapt the path 4-tuple to the one of the last packet received, and
   next packets on path 1 will use that 4-tuple. Because the Path ID is
   its only identifier, a path is not bound to a particular 4-tuple, and
   can shift to another one. Multipath QUIC thus integrates the
   Connection Migration ability at path level, providing the Path
   Migration ability.

3.9. Coping with Address Removals

   During the life of the QUIC connection, an 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 REMOVE_ADDRESS
   frames. Those frames contain the Address ID of the lost address.

   Thanks to this frame, an host can stop using a path whose 4-tuple
   contains the removed address. However, if a NAT'd client losses its
   private address and advertises this event, the sever will not know to
   which public address this advertisement is intended and might still

   use paths using that particular address. The PATHS frame helps the
   server to perform the mapping between both addresses. It contains
   for each active path the local Address ID used by the sending host.
   With both the PATHS and the REMOVE_ADDRESS frames, the server can
   identify which paths will be affected by the address removal to stop
   their usage before being migrated to another 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. Packet Format Changes

   This section describes the changes required in the packet format. As
   previously described, multipath capabilities must be enabled after
   connection establishment. Because the initial path has Path ID 0,
   packets with long headers are considered to be sent on Path 0,
   therefore no change is required on the wire. Since only short
   headers are expected to be sent once the connection is established,
   only those require the Path ID as a field. The proposed short header
   is presented in Figure 5.

```
 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|C|K|M|Type(4)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                     [Connection ID (64)]                      +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| [Path ID (8)] |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Packet Number (8/16/32)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Protected Payload (*)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
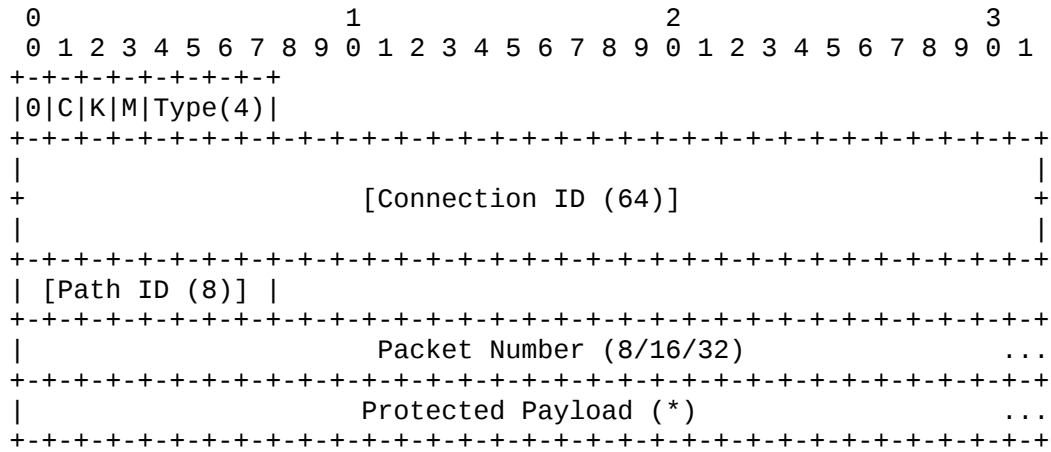
                 Figure 5: New Short Header for Packets

   The usage of multiple paths is indicated with the M bit. If this bit
   is set, the packet also contains the Path ID field indicating the
   path the packet belongs to. If the multipath flag is not set, the
   packet belongs to path 0. Notice that Path ID 0 can be used as a
   regular path to exchange data, but a single-path connection MUST use
   path 0 for all exchanged packets.

5. Using Multiple Paths

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 the connection is the
   minimum between both advertised values. A connection can then use
   any path with a Path ID comprised between 0 and the negotiated
   maximum path ID inclusive. If one of the host does not provides 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. Packets with a Path ID greater
   than the negotiated maximum value MUST be ignored.

5.1.1. Transport Parameter Definition

   An endhost MAY use the following transport parameter:

   max_path_id (0x0007): The maximum path ID transport parameter
      indicates the number of paths in addition to the Initial Path that
      the host is willing to use over the connection, encoded as an
      unsigned 8-bit integer. This indicates that received packets
      larger than this limit will be dropped. The default value for
      this parameter is 0, meaning that an host omitting this transport
      parameter does not want to use multiple paths over the connection.

5.2. Path State

   A path is associated to a UDP flow over which packets can be sent or
   received. The following state is maintained for a given path.

   o Path ID: this 1-byte number uniquely identifies a path in a
     connection. This value is immutable.
   o Packet Number Space: each path is associated with own monotonically
     increasing packet number space. Each endpoint maintains a separate
     packet number for sending and receiving. Packet number
     considerations described in {{I-D.ietf-quic-transport}} apply
     within a given path.
   o Current 4-tuple: the tuple (sIP, dIP, sport, dport) used by the
     path to send or receive packets. This value is mutable and can also
     be empty when the path is not in use. If this value is set, the
     path is considered as active. The tuple can change either because
     the host decides to change its local address and/or port, or
     because it receives a packet with a different remote address and/or
     port than currently recorded. To cope with possible packet
     reordering within a given path, the remote address and port
     recorded by the host MUST match the one of the received packet with
     the largest Packet Number. The Initial Path MUST remain active at
     any time of the connection.
   o Current (local Address ID, remote Address ID) tuple: those
     identifiers come from the ADD_ADDRESS sent (local) and received
     (remote). This enables host to detect a path as unusable when,
     e.g., the remote Address ID is declared as lost by a
     REMOVE_ADDRESS. The addresses on which the connection was
     established have Address ID 0. The reception of PATHS frames
     helps hosts to perform the matching between the remote Address ID
     and the path.
   o 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 an host needs to perform
     packet scheduling decisions and flow control management.

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. The only exception is the ACK frame that
   contains packet-level information with the Largest Acknowledged
   field. Since the Packet Number are now linked to paths, the ACK
   frame must contain the Path ID it acknowledges.

6.1. ACK Frame

   This draft changes the type byte of the ACK frame. The new type is
   formatted as "10PNLLMM", and is parsed as follows:

   o The two high-order bits must be set to 10 indicating that this is
   an ACK frame. This changes from the 101 prefix of the current QUIC
   design [I-D.ietf-quic-transport]. Note that the prefix 100 is
   currently unused in [I-D.ietf-quic-transport].

   o The "P" bit indicates the presence of the Path ID field in the ACK
   frame. If unset, the ACK frame relates to the default path (i.e.,
   Path ID 0).

   o Other bits remain unchanged.

   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 (8)] |[Num Blocks(8)]|   NumTS (8)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Largest Acknowledged (8/16/32/48)          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        ACK Delay (16)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    ACK Block Section (*)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Timestamp Section (*)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
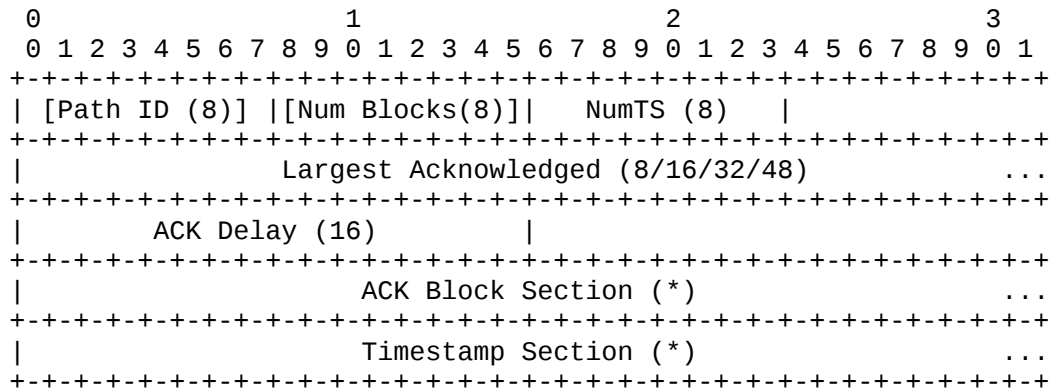
                  Figure 6: ACK Frame adapted to Multipath

   Compared to the ACK frame in the current QUIC design
   [I-D.ietf-quic-transport], the ACK frame can contain an optional Path
   ID field indicating to which path the acknowledged PSNs relate to.
   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. The impact of such a strategy on the latency estimation has to
be explored further.

7. New Frames

To support the multipath operations, new frames have been defined to
coordinate hosts. This draft uses a type field containing 0x10 to
indicate that the frame is related to multipath operations.

7.1. ADD_ADDRESS Frame

The ADD_ADDRESS frame is used by a host to advertise its currently
reachable addresses. The proposed type for the ADDRESS frame is
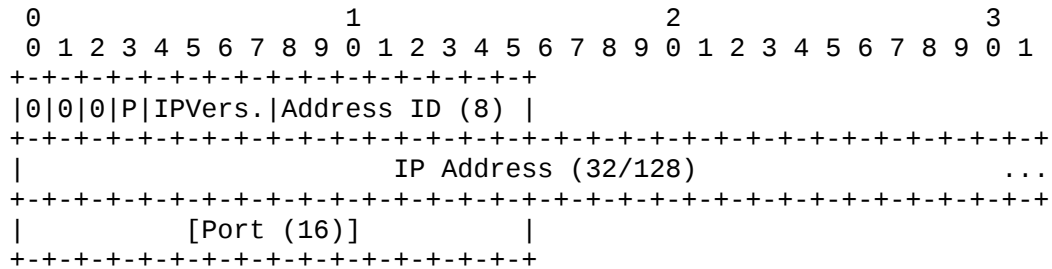0x10. An ADD_ADDRESS frame is shown below.

```
 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 7: 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 contains the version of the IP address contained in the IP
  Address field.

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).

7.2. 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 0x11. 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) |
+-+-+-+-+-+-+-+-+
```
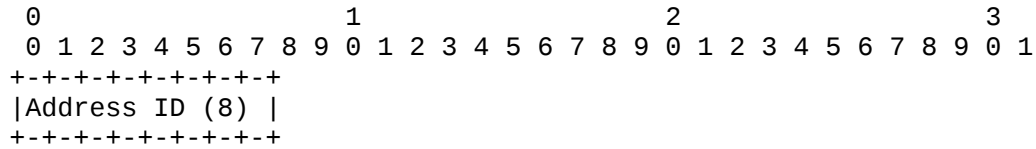
                     Figure 8: REMOVE_ADDRESS Frame

   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 until they have been migrated to another available
   address. If the REMOVE_ADDRESS contains an Address ID that was not
   previously announced, host MUST ignore the frame.

7.3. 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.
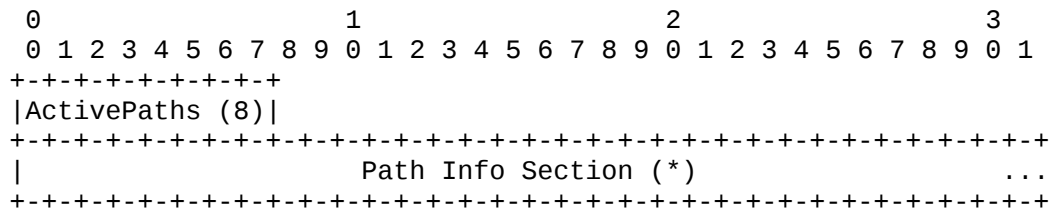   Its proposed type is 0x12. The format of the PATHS frame is shown
   below.

```
 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
+-+-+-+-+-+-+-+-+
|ActivePaths (8)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Path Info Section (*)                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                        Figure 9: PATHS Frame

   The PATHS frame contains the following fields:

   o ActivePaths: the current number of paths considered as active
     sender point of view, excluding the Initial Path. ActivePaths MUST
     be lower or equal to the maximum Path ID negotiated.

   o Path Info Section: contains information about all the active paths
     (i.e., there are ActivePaths + 1 entries). The format of this
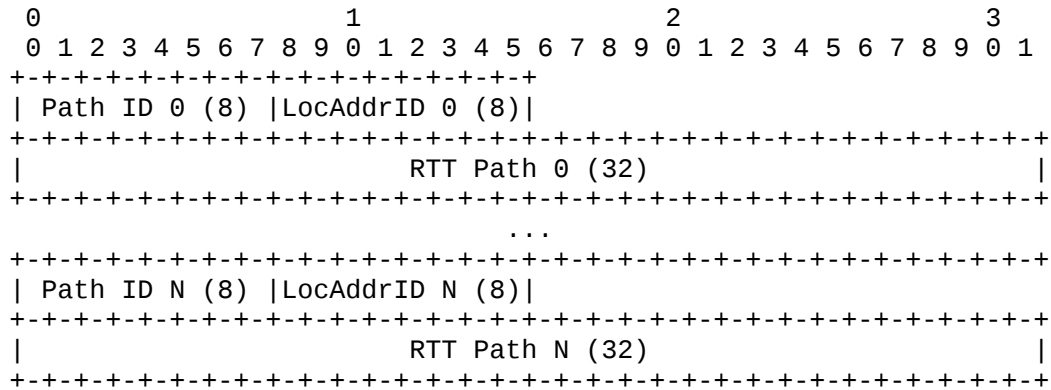   section is shown below.

```
 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 (8) |LocAddrID 0 (8)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          RTT Path 0 (32)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Path ID N (8) |LocAddrID N (8)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          RTT Path N (32)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                       Figure 10: 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.

   o RTT Path: the RTT experienced by the sending host over the path
     with the provided Path ID. The formatting is similar to the one
     used for the ACK delay field in the ACK frame.

   The Path Info section currently contains the RTT of the sending host,
   but this section can be extended to provide additional information in
   order to get a global picture of the connection at both ends.

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 follow. 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 8 bits of the Path ID 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.

9. IANA Considerations

9.1. QUIC Transport Parameter Registry

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

```
+--------+----------------+---------------+
| Value  | Parameter Name | Specification |
+--------+----------------+---------------+
| 0x0007 | max_path_id    | Section 5.1.1 |
+--------+----------------+---------------+
```

           Table 1: Addition to QUIC Transport Parameters Entries

10. References

10.1. Normative References

   [I-D.ietf-quic-recovery]
              Iyengar, J. and I. Swett, "QUIC Loss Detection and
              Congestion Control", draft-ietf-quic-recovery-06 (work in
              progress), September 2017.

   [I-D.ietf-quic-tls]
              Thomson, M. and S. Turner, "Using Transport Layer Security
              (TLS) to Secure QUIC", draft-ietf-quic-tls-07 (work in
              progress), October 2017.

   [I-D.ietf-quic-transport]
              Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed
              and Secure Transport", draft-ietf-quic-transport-07 (work
              in progress), October 2017.

   [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>.

10.2. 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.

   [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.

Authors' Addresses

   Quentin De Coninck
   UCLouvain

   Email: quentin.deconinck@uclouvain.be

   Olivier Bonaventure
   UCLouvain

   Email: Olivier.Bonaventure@uclouvain.be