QUIC
Internet-Draft
Intended status: Standards Track
Expires: 9 September 2021

Y. Liu
Y. Ma
Alibaba Inc.
C. Huitema
Private Octopus Inc.
Q. An
Alibaba Inc.
Z. Li
ICT-CAS
8 March 2021

Multipath Extension for QUIC
draft-liu-multipath-quic-03

Abstract

This document specifies multipath extension for the QUIC protocol to enable the simultaneous usage of multiple paths for a single connection. The extension is compliant with the single-path QUIC design. The design principle is to support multipath by adding limited extension to [QUIC-TRANSPORT].

Status of This Memo

Copyright Notice

Table of Contents

1. **Introduction**

In this document, we propose an extension to the current QUIC design to enable the simultaneous usage of multiple paths for a single connection.

This proposal is based on several basic design points:

* Re-use as much as possible mechanisms of QUIC-v1, which has supported connection migration and path validation.

* To avoid the risk of packets being dropped by middleboxes (which may only support QUIC-v1), use the same packet header formats as QUIC V1.

* Endpoints need a Path Identifier for each different path which is used to track states of packets. As we want to keep the packet header formats unchanged [QUIC-TRANSPORT], Connection IDs (and the sequence number of Connection IDs) would be a good choice of Path Identifier.

* For the convenience of packet loss detection and recovery, endpoints use a different packet number space for each Path Identifier.

* Congestion Control, RTT measurements and PMTU discovery should be per-path (following [QUIC-TRANSPORT])

This document is organized as follows. It first provides definitions of multipath quic in Section 2. It then specifies how to enable multipath quic during handshake in Section 3, and path management in Section 4. It discusses packet scheduling in Section 7, and congestion control in Section 8. The new frames are defined in Section 9.

2. Conventions and Definitions

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

We assume that the reader is familiar with the terminology used in [QUIC-TRANSPORT]. In addition, we define the following terms:

* Path Identifier: An identifier that is used to identify a path in a QUIC connection at an endpoint. It is defined as the sequence number of the destination Connection ID used for sending packets on that particular path.

* Each node maintains a list of "Received Packets" for each of the CID that it provided to the peer, which is used for acknowledging packets received with that CID.

3. Enable Multipath QUIC - Handshake

This extension defines a new transport parameter, used to negotiate the use of the multipath extension during the connection handshake, as specified in [QUIC-TRANSPORT]. The new transport parameter is defined as follow:

* name: enable_multipath (TBD - experiments use 0xbaba)

* value: 0 (default) for disabled, 1 for enabled

If the peer does not carry the enable_multipath(TBD - experiments use 0xbaba) transport parameter, which means the peer does NOT support multipath, endpoint MUST fallback to [QUIC-TRANSPORT] with single path and MUST NOT send any MP frames in the following packets, also MUST NOT use the multipath specific AEAD algorithm defined in Section 5.1.

Notice that transport parameter "active_connection_id_limit" [QUIC-TRANSPORT] limits the number of usable Connection IDs, and also limits the number of concurrent paths.

4. Path Management

After endpoints have negotiated in handshake flow that both endpoints enable multipath feature, endpoints can start using multiple paths.

This proposal add one frame for path management:

* PATH_STATUS frame for the receiver side to claim the path state and preference

All the new MP frames are sent in 1-RTT packets [QUIC-TRANSPORT].

4.1. Path Identifier and Connection ID

Endpoints need a Path Identifier for each different path which is used to track states of packets. Endpoints use Connection IDs in 1-RTT packet header as Path Identifier in each directions, and use the sequence number of Connection IDs in MP frames to identify the path referred.

Following [QUIC-TRANSPORT], Each endpoint uses NEW_CONNECTION_ID frames to claim usable connections IDs for itself. Before an endpoint add a new path, it SHOULD check whether there is at least one unused available Connection ID for each side.

Endpoints can find which path a received packet belongs to according to the Destination Connection ID of the 1-RTT packet. Endpoints can find the context of a path by its' Connection ID or the Sequence number of Connection ID.

4.2. Path Packet Number Spaces

For the convenience of packet loss detection and recovery, endpoints

use a different packet number space for each Path Identifier
(Connection ID). ACK_MP frame includes the sequence number of the
Destination Connection ID of the acknowledged packets as the Path
Identifier.

4.3. Path Initiation

Figure 1 illustrates an example of new path establishment.

```
Client                                                  Server

(Exchanges start on default path)
1-RTT[]: NEW_CONNECTION_ID[C1, Seq=1] -->
                      <-- 1-RTT[]: NEW_CONNECTION_ID[S1, Seq=1]
                      <-- 1-RTT[]: NEW_CONNECTION_ID[S2, Seq=2]
...
(starts new path)
1-RTT[0]: DCID=S2, PATH_CHALLENGE[X] -->
                  Checks AEAD using nonce(CID sequence 2, PN 0)
    <-- 1-RTT[0]: DCID=C1, PATH_RESPONSE[X], PATH_CHALLENGE[Y],
                                             ACK_MP[Seq=2,PN=0]
Checks AEAD using nonce(CID sequence 1, PN 0)
1-RTT[1]: DCID=S2, PATH_RESPONSE[Y],
          ACK_MP[Seq=1, PN=0], ... -->
```

Figure 1: Example of new path establishment

As shown in Figure 1, client provides one unused available Connection
ID (C1 with sequence number 1), and server provides two available
Connection IDs (S1 with sequence number 1, and S2 with sequence
number 2). When client wants to start a new path, it checks whether
there is unused available Connection IDs for each side, and choose an
available Connection ID S2 as the Destination Connection ID in the
new path.

Endpoints need to exchange unused available Connection IDs with the
NEW_CONNECTION_ID frame before an endpoint starts a new path. For
example, if the goal is to maintain 2 paths, each endpoint should
provide at least 3 CID to its peer: 2 in use, and one spare. If the
client has used all the allocated CID, it is supposed to retire those
that are not used anymore, and the server is supposed to provide
replacements, as specified in [QUIC-TRANSPORT].

If the transport parameter "active_connection_id_limit" is negotiated
as N, and the server has provided N Connection IDs and the client has
started N paths, the limit is reached. If the client wants to start
a new path, it has to retire one of the established paths.

Path validation uses the PATH_CHALLENGE and PATH_RESPONSE frame
defined in QUIC-Transport [QUIC-TRANSPORT].

4.4. Path State Management

An endpoint uses PATH_STATUS frames to inform that the peer should send packets in the preference expressed by these frames. An endpoint uses the sequence number of the CID used by the peer for PATH_STATUS frames (describing the sender's path identifier).

In the example Figure 1, if the client wants to send a PATH_STATUS frame to tell the server that it prefers the path with CID sequence number 1 (of the server's side), the client should use the identifier of the server (sequence 1) in PATH_STATUS frame.

PATH_STATUS frame describes 4 kinds of path states:

* Abandon a path, and release the corresponding resource.

* Mark a path as "available", i.e., allow the peer to use its own logic to split traffic among available paths.

* Mark a path as "standby", i.e., suggest that no traffic should be sent on that path if another path is available.

* Mark the priority of a path, i.e, path 1 is weight 8, path 2 is weight 2, suggest that path 1 has higher priority than path 2, and peer should try to send more data in path 1.

PATH_STATUS frame can be sent via a different path, instead of the path identified by the Path Identifier field.

4.5. Path Close

An endpoint that want to delete a path SHOULD NOT rely on implicit signals like idle time or packet losses, but instead SHOULD use explicit ask to abandon path by sending the PATH_STATUS frame.

4.5.1. Use PATH_STATUS frame to close a path

Both client and server can close a path, by sending PATH_STATUS frame which abandons the path with a corresponding Path Identifier. Once a path is marked as "abandon", it means that the resources related to the path can be released.

Figure 2 illustrates an example of path closing. In this case, we are going to close the first path. For the first path, the server's 1-RTT packets use DCID C1, which has a sequence number of 1; the client's 1-RTT packets use DCID S2, which has a sequence number of 2. For the second path, the server's 1-RTT packets use DCID C2, which has a sequence number of 2; the client's 1-RTT packets use CID S3, which has a sequence number of 3. Note that two paths use different packet number space. (For the convience of distinguishing the CID sequence number and PATH_STATUS sequence number, we call the "PATH_STATUS sequence number" as "PSSN".)

```
Client                                                         Server

(client tells server to abandon a path)
1-RTT[X]: DCID=S2 PATH_STATUS[id=1, PSSN1, status=abandon, pri.=0] ->
                               (server tells client to abandon a path)
<- 1-RTT[Y]: DCID=C1 PATH_STATUS[id=2, PSSN2, status=abandon, pri.=0],
                                                   ACK_MP[Seq=2, PN=X]
(client abandons the path that it is using)
1-RTT[U]: DCID=S3 RETIRE_CONNECTION_ID[2], ACK_MP[Seq=1, PN=Y] ->
                           (server abandons the path that it is using)
     <- 1-RTT[V]: DCID=C2 RETIRE_CONNECTION_ID[1], ACK_MP[Seq=3, PN=U]
```

Figure 2: Example of closing a path

In scenarios such as client detects the network environment change (client's 4G/Wi-Fi is turned off, Wi-Fi signal is fading to a threshold), or endpoints detect that the quality of RTT or loss rate is becoming worse, client or server can terminate a path immediately.

4.5.2. Effect of RETIRE_CONNECTION_ID frame

Receiving a RETIRE_CONNECTION_ID frame causes the endpoint to discard the resources associated with that connection ID. If the connection ID was used by the peer to identify a path from the peer to this endpoint, the resources include the list of received packets used to send acknowledgements. The peer MAY decide to keep sending data using the same IP addresses and UDP ports previously associated with the connection ID, but MUST use a different connection ID when doing so.

4.5.3. Idle timeout

[QUIC-TRANSPORT] allows for closing of connections if they stay idle for too long. The connection idle timeout in multipath QUIC is defined as "no packet received on any path for the duration of the idle timeout". It means that if all paths remain idle for the idle timeout, the connection is implicitly closed.

5. Using TLS to Secure QUIC Multipath

In order to facilitate loss detection and recovery when sending data over multiple paths, this specification defines how packets sent over multiple paths use different packet number spaces. This requires changes in the way AEAD is applied for packet protection, as explained in Section 5.1, and tighter constrainst for key updates, as explained in Section 5.2.

5.1. Packet protection for QUIC Multipath

Packet protection for QUIC V1 is specified is section 5 of [QUIC-TLS]. The general principles of packet protection are not changed for QUIC Multipath. No changes are needed for setting packet protection keys, initial secrets, header protection, use of 0-RTT keys, receiving out-of-order protected packets, receiving protected packets, or retry packet integrity. However, the use of multiple number spaces for 1-RTT packets requires changes in AEAD usage.

Section 5.3 of [QUIC-TLS] specifies AEAD usage, and in particular the use of a nonce, N, formed by combining the packet protection IV with the packet number. QUIC multipath uses multiple packet number spaces, and thus the packet number alone would not guarantee the uniqueness of the nonce. In order to guarantee this uniqueness, we construct the nonce N by combining the packet protection IV with the packet number and with the identifier of the path, which for 1-RTT packets is the Sequence Number of the Destination Connection ID present in the packet header, as defined in Section 5.1.1 of [QUIC-TRANSPORT], or zero if the Connection ID is zero-length. Section 19 of [QUIC-TRANSPORT] encode this Connection ID Sequence Number as a A variable-length integer, allowing values up to 2^62-1; for QUIC multipath, we require that a range of no more than 2^32-1 values be used without updating the packet protection key.

For QUIC multipath, the construction of the nonce starts with the construction of a 96 bit path-and-packet-number, composed of the 32 bit Connection ID Sequence Number in byte order, two zero bits, and the 62 bits of the reconstructed QUIC packet number in network byte order. If the IV is larger than 96 bits, path-and-packet-number is left-padded with zeros to the size of the IV. The exclusive OR of the padded packet number and the IV forms the AEAD nonce.

For example, assuming the IV value is "6b26114b9cba2b63a9e8dd4f", the connection ID sequence number is "3", and the packet number is "aead", the nonce will be set to "6b2611489cba2b63a9a873e2".

5.2. Key Update for QUIC Multipath

The Key Phase bit update process for QUIC V1 is specified in Section 6 of [QUIC-TLS]. The general principles of key update are not changed for Multipath QUIC. Following QUIC V1, the Key Phase bit is used to indicate which packet protection keys are used to protect the packet. The Key Phase bit is toggled to signal each subsequent key update. Because of network delays, packets protected with the older key might arrive later than the packets protected with the new key. Therefore, the endpoint needs to retain old packet keys to allow these delayed packets to be processed and it must distinguish between the new key and the old key. In QUIC V1, this is done using packet numbers so that the rule is made simple: Use the older key if packet number is lower than any packet number frome the current key phase.

In QUIC multipath, some care is needed in the initiating Key Update process. Because different paths use different packet number spaces but share a single key, when a key update is initiated on one path, packets sent to the other path needs to know when transition is complete. Otherwise, it is possible that the other paths send packets with the old keys, but skip sending any packets in the current key phase and directly jump to sending packet in the next key phase. When that happens, as the endpoint can only retain two sets of packet protection keys with the 1-bit Key Phase bit, the other paths cannot distinguish which key should be used to decode received packets, which results in a key rotation synchronization problem.

To address such a synchronization issue, in QUIC multipath, if key update is initilized on one path, the sender should send at least one packet with the new key on all active paths. Regarding the responding to Key Update process, the endpoint MUST NOT initiate a subsequent key update until a packet with the current key has been acknowledged on each path.

Following the Section 5.4. of [QUIC-TLS], the Key Phase bit is protected, so sending multiple packets with Key Phase bit flipping at the same time should not cause linkability issue.

6. Using Multipath QUIC with load balancers

This specification follows the Connection ID negotiation defined in [QUIC-TRANSPORT]. For stateless or low-state load balancers supporting Multipath QUIC, implementations SHOULD use the specification of Connection ID generation and Load balancer routing defined in [QUIC-LB], guarantee that packets with Connection IDs belonging to the same connection, can be routed to same server.

7. Packet scheduling

7.1. Basic Scheduling

For an outgoing packet, the packet scheduler decides which path the packet shall be transmitted. A basic static scheduling strategy consists of four major components:

1. Path state: A scheduler may want to decide which path shall be activated to transmit data. For instance, a scheduler can choose to use only one of the two paths and completely ignore the other one. A scheduler marks the selected paths to be in the "available" state and the un-selected ones in the "standby" state.

2. Path priority: Due to the fact that costs of transmitting data over different paths are not always equal. For example, the energy (battery) cost over a 5G path and a wifi path are very different. In another example, transmissions over a wifi path and a cellular path may incur different charges per packet. Note that a user's preference may change over time. For instance, certain mobile carriers offer unlimited free data for a

particular streaming app. Therefore, the path priority should be made available in the scheduler.

3. Path selection algorithm: A selection algorithm splits packets across different paths and determines the order of paths to be selected. The selection algorithm takes congestion controller states as inputs, such as smoothed RTTs (sRTTs), estimated bandwidths (eBWs) and congestion window sizes (CWNDs) as well as application-defined information such as path priorities and path states. The outputs of the algorithm is an ordered list of paths to put a packet on. To name a few, some of the commonly used algorithms are: - Round-Robin: There is no priority. it selects paths one by one in order to transmit data. - Lowest-RTT: It first chooses the path with the lowest RTT and feeds packets to it until that path's congestion window is full. Then it chooses the path with the second lowest RTT. - Highest-Sending-Rate: It first chooses the path with the highest bandwidth and feeds packets to it until that path's congestion window is full. Then it chooses path with the second largest bandwidth.

4. Packet redundancy: One major challenge in multi-path transmission is that a packet loss on the slow path might block the overall transmission when packets are split across fast-changing paths. As the path selection algorithm takes inputs from congestion controllers on predictions of the network which may not be accurate enough for fast-changing wireless channels, such an imprecise estimation could lead to network overuse/underuse. A solution to this problem is to implement packet redundancy strategy. A redundancy strategy can be applied to only ACK packets(partial redundancy) or all data packets (full redundancy). It is up to the application to determine whether, when, and on which packets to activate redundancy.

The path state and path priority are managed by PATH_STATUS frame. The path selection algorithm and packet redundancy are application related and should be controlled by the applicaiton.

7.2. Scheduling with QoE Feedback

Applications may have completely different QoE requirements---the interactive applications are delay sensitive, while the video streaming applications are more throughput sensitive. There is thus a trend of cross-layer design that takes applications' demands into account when managing paths or scheduling packets. The QoE feedback is used to fully support application-awareness in multipath scheduling and is carried in the QOE_CONTROL_SIGNALS frames Figure 6. The QOE_CONTROL_SIGNALS frames can include general application-level information that is needed by the schedulers. The frequency of such feedback should be controlled to limit the amount of extra packets. The QoE control signal allows a synchronization of viewpoints between two endhosts. It is up to the application to determine the interpretation of QoE control signals.

7.3. Per-stream Policy

As QUIC supports stream multiplexing, streams are allowed to associate stream priorities to express applications intent. For instance, objects in a web page may be dependent on others and thus have different priorities multipath quic scheduler. A stream priority-aware packet scheduling algorithm will improve the performance notably.

```
    High priority  /\  +---------+
                   ||  |         |
                   ||  +---------+
                   ||  +---------+
                   ||  |         |
                   ||  +---------+
                   ||     ...         User-defined stream priority
                   ||  +---------+
    Low priority   ||  |         |
                   ||  +---------+
    -----------------------------------------------------------
    High priority  /\  +---------+
                   ||  |         |
                   ||  +---------+
                   ||  +---------+
                   ||  |         |
                   ||  +---------+
                   ||     ...         Default stream priority
                   ||  +---------+
    Low priority   ||  |         |
                   ||  +---------+
```

Figure 3: Stream priority

The priority management scheme composes two separated priority ranges. The user-defined priority range includes those streams that the applications explicitly designate priorities, while the default priority range includes the streams with no priorities set by the applications. Only when the streams in the user-defined ranges have no data to send, the streams in the default priority range can send. In the same range, one can use the weighted-round robin for scheduling---the higher-priority streams get more quota for data to send in each round. One can also dynamically set/change the priorities of the streams in the default priority ranges to enable short stream first if needed.

8. Congestion control and loss detection

8.1. Congestion control

Implementations MAY support coupled congestion controllers such as LIA [MPTCP-LIA], OLIA [MPTCP-OLIA], and etc., or support decoupled congestion controllers in environments using disjoint network paths.

In decoupled congestion control, each path runs its own congestion controller without interacting with the congestion controllers of other paths. That is to say, in the aspect of congestion control, a path behaves exactly the same as a normal QUIC connection over the same network path.

Each path MAY choose congestion control algorithm independently.

8.2. Packet number space and acknowledgements

Each path has it's own packet number space for transmitting 1-RTT packets.

Acknowledgements of Initial and Handshake packets MUST be carried using ACK frames, as specified in [QUIC-TRANSPORT]. The ACK frames, as defined in [QUIC-TRANSPORT], do not carry path identifiers. If for some reason ACK frames are received in 1RTT packets while the state of multipath negotiation is ambiguous, they MUST be interpreted as acknowledging packets sent on path number 0. After endpoints successfully negotiate multipath support, they SHOULD use ACK_MP frames instead of ACK frames to signal acknowledgement of 1-RTT packets, and also 0-RTT packets as specified in Section 10.1.

ACK_MP frame Section 9.2 can be returned via either a different path, or the same path identified by the Path Identifier, based on different strategies of sending ACK_MP frames.

8.3. Flow control

TBD.

9. New frames

All the new frames MUST be sent in 1-RTT packet, and MUST NOT use other encryption levels.

If an endpoint receives MP frames from packets of other encryption levels, it MUST return MP_PROTOCOL_VIOLATION as a connection error and close the connection.

9.1. PATH_STATUS frame

PATH_STATUS Frame are used by endpoints to inform the peer of the current status of one path, and the peer should send packets according to the preference expressed in these frames. Endpoint use the sequence number of the CID used by the peer for PATH_STATUS frames (describing the sender's path identifier). PATH_STATUS frames are formatted as shown in Figure 4.

```
  PATH_STATUS Frame {
    Type (i) = TBD-03 (experiments use 0xbaba03),
    Path Identifier (i),
    Path Status sequence number (i),
    Path Status (i),
    Path Priority (i),
  }
```

Figure 4: PATH_STATUS Frame Format

PATH_STATUS Frames contain the following fields:

Path Identifier: A variable-length integer specifying the path identifier.

Path Status sequence number: A variable-length integer specifying the sequence number assigned for this PATH_STATUS frame. There is a different path status sequence number space for each path.

Available values of Path Status field are:

* 0: Abandon

* 1: Standby

* 2: Available

If the value of Path Status field is 2-available, the receiver side can use the Path Priority field to express the priority weight of a path for the peer.

Frames may be received out of order. A peer MUST ignore an incoming PATH_STATUS frame if it previously received another PATH_STATUS frame for the same Path Identifier with a sequence number equal to or higher than the sequence number of the incoming frame.

PATH_STATUS frames SHOULD be acknowledged. If a packet containing a PATH_STATUS frame is considered lost, the peer should only repeat it if it was the last status sent for that path -- as indicated by the sequence number.

9.2. ACK_MP frame

ACK_MP frame allows for acknowledgements on different paths. ACK_MP frame is formatted by adding a Path Identifier field to [QUIC-TRANSPORT] ACK frame. ACK_MP frame is formatted as shown in Figure 5.

```
  ACK_MP Frame {
    Type (i) = TBD-00..TBD-01 (experiments use 0xbaba00..0xbaba01),
    Path Identifier (i),
    Largest Acknowledged (i),
    ACK Delay (i),
    ACK Range Count (i),
    First ACK Range (i),
    ACK Range (..) ...,
    [ECN Counts (..)],
  }
```

Figure 5: ACK_MP Frame Format

Type(i) = TBD-00 (experiments use 0xbaba00) , with no ECN Counts
Type(i) = TBD-01 (experiments use 0xbaba01) , with ECN Counts

9.3. QOE_CONTROL_SIGNALS frame

QOE_CONTROL_SIGNALS frame is used to carry quality of experience (QoE) information. A typical use of such information is to provide feedback to help application-aware scheduling. Note that different applications may have very different needs, the interpretation of the QoE control signal can be up to the users. QOE_CONTROL_SIGNALS frames are formatted as shown in Figure 6.

```
  QOE_CONTROL_SIGNALS Frame {
    Type (i) = TBD-02 (experiments use 0xbaba02),
    Path Identifier (i),
    QoE Control Signals Length(8),
    QoE Control Signals (..)
  }
```

Figure 6: QOE_CONTROL_SIGNALS Frame Format

QOE_CONTROL_SIGNALS frames may be received out of order, peers SHOULD pass them to the application as they arrive. Although QOE_CONTROL_SIGNALS frames are not retransmitted upon loss detection, they are ack-eliciting [QUIC-RECOVERY].

10. Implementation Considerations

Management of acknowledgements delay If implementation uses ACK_FREQUENCY Frame in [QUIC-DELAYED-ACK] to let senders control the frequency of acknowledgements, the same mechanism can be used in multi-path QUIC. There are two parameters in the ACK_FREQUENCY Frame, "Packet Tolerance" and "Update Max Ack Delay".

Those two parameters are typically computed in real time based on observed performance:

* "Packet Tolerance" is set to a fraction of the congestion window

* "Update Max Ack Delay" is set to a fraction of the RTT -- but not smaller than the specified min delay

In multi-path QUIC, there are multiple paths with different RTT and different congestion windows. In this draft, it is suggested that implementations can use the smallest RTT of the available paths to compute the delay, and use the sum of congestion windows of all available(not including standby/abandon state) paths.

10.1. Handling of 0-RTT packets

The draft specifies a packet number space for each path. Because multi-path is enabled after the handshake negotiation complete, there will be a separate context for each Connection ID after multi-path is negotiated. 0-RTT packets are sent before these per path contexts are established. To avoid confusion, this draft provides a way for implementations to deal with 0-RTT packets that is both easy to implement and compatible with [QUIC-TRANSPORT]:

* All 0-RTT packet are initially tracked in the "global" application context.

* On the client side, 0-RTT packets are initially sent in the "global" application context. The handshake concludes before any 1-RTT packet can be sent or received. When the handshake completes, if multipath is negotiated, the tracking of 0-RTT packets moves from the "global" application context to the "path 0" application context. That means the sequence number of the first 1-RTT packets sent by the client will follow the sequence number of the last 0-RTT packet.

* On the server side, the negotiation completes after the client first flight is received and the the server first flight is sent. 0-RTT packets are received after that. If multipath is negotiated, they are considered received on "path 0".

In conclusion, 0-RTT packets are tracked and processed with path identifier 0.

11. Security Considerations

TBD.

12. IANA Considerations

This document defines a new transport parameter for the negotiation of enable multiple paths for QUIC, and three new frame types. The draft defines provisional values for experiments, but we expect IANA to allocate short values if the draft is approved.

The following entry in Table 1 should be added to the "QUIC Transport Parameters" registry under the "QUIC Protocol" heading.

Value	Parameter Name.	Specification

TBD (experiments use 0xbaba)	enable_multipath	Section 3

Table 1: Addition to QUIC Transport Parameters Entries

The following frame types defined in Table 2 should be added to the "QUIC Frame Types" registry under the "QUIC Protocol" heading.

Value	Frame Name	Specification
TBD-00 - TBD-01 (experiments use 0xbaba00-0xbaba01)	ACK_MP	Section 9.2
TBD-02 (experiments use 0xbaba02)	QOE_CONTROL_SIGNALS	Section 9.3
TBD-03 (experiments use 0xbaba03)	PATH_STATUS	Section 9.1

Table 2: Addition to QUIC Frame Types Entries

13. Changelog

14. Appendix.A Scenarios related to migration

In QUIC V1, there are four scenarios related to migration: CID renewal, NAT Rebinding, controlled migration, and migration to server preferred address. It would be useful to explain exactly how these four scenarios are supported or changed with Multipath QUIC. For V1, these scenarios are described as follow:

* CID Renewal happens when the client starts using a new CID for 1-RTT packet, while still using the same four-tuple. This is typically done for privacy, for example after a long period of silence. The expected result is that the server will also use a new CID for its next packets. In that scenario, RTT and congestion control parameters remain the same before and after migration.

* NAT Rebinding happens when a NAT on the path changes its mappings. The server receives packets that bear the same CID as previously, but arrive on a different four tuple. The complication is that this could be an attack in which the attacker captures a packet from the client and resends it from a different address. The server is expected to perform continuity tests for both the old

and the new path, typically using a different CID for the new path. If the continuity test on the new path succeeds before the old path, the server migrates to the new path, otherwise it continues using the old path and ignores the new path.

* Controlled migration happens when a client tests a new path. The server receives packets that bear a new CID and arrive on a new four tuple. The server responds to the path challenge, perform its own continuity test on the new path. If the client sends non-path-validation packets on the new path, the server switches to sending on the new path and discards the old path.

* Preferred address migration happens when the server sends the preferred address TP during the exchange. The client performs a controlled migration to the new path, and if that is successful discards the old path.

We could sum up these scenarios in the following table:

CID	4-tuple	preferred address	result
Old	Old	-	Not a migration.
Old	New	-	NAT Rebinding.
New	Old	-	CID Renewal.
New	New	matches PFA	Migration to Preferred Address.
New	New	other	Controlled Migration.

Table 3: Scenarios related to migration

The expectation in those scenarios is:

Scenario	Expectation
Not a migration	Continue using existing path
NAT Rebinding	After validation, use new path and discard previous path.
CID Renewal	Create new path with new CIDs, discard old path. Reuse RTT and CC parameter.

Controlled Migration	Create new path with new CIDs. Server creates a new path,ready to use both paths. Client may later discard old path.
Migration to Preferred Address	Same as Controlled Migration, but the client is expected to abandon the old path

Table 4: Expectation in scenarios related to migration

In multipath quic, client / server create a new path and abandon the old path to do exactly the same thing as connection migration in the previous scenarios.

15. Appendix.B Considerations on RTT estimate and loss detection

QUIC implementations use RTT estimates in many ways:

* For loss detection, RTT estimates are used to evaluate how long to wait for an acknowledgement before a packet is declared lost.

* Several congestion control algorithm (e.g. LEDBAT, VEGAS, HYSTART) use variations of the RTT above the minimum value to detect the beginning of congestion.

* BBR uses the minimal RTT to compute the minimal size of the congestion window for a target data rate.

* ACK delays are often set as a fraction of the RTT.

In a multipath environment, the RTT can be estimated each time a new packet is acknolwedged. However, the observed RTT will vary not only based on the state of the send path, but also based on the choice of the return path used for acknowledgements. Each RTT measurement will the sum of the one-way delay on the send path and the one-way delay on the return path. This has a number of implications for the different ways of using the RTT presented above:

* If the goal is to detect possible losses, it is probably sufficient to consider all RTT measurements for a given path. Classic formulas like adding smoothed RTT and a number of deviations aim at estimating a reasonable upper bound of the acknowledgement delays. Statistics on observed acknowledgement delays will provide a valid estimate, regardless of the selection of the return path by the peer.

* If the goal is to detect the onset of collision and tune a congestion algorithm, the variations of delays due to the choices of return paths will be a source of errors. Implementations will need to pick a strategy, such as for example only considering acknowledgements received through the "fastest" return path, or

maybe those received through the matching four tuple for the sending path. An alternative would be to use time stamps to directly estimate variations of the one way delays. [QUIC-Timestamp] provides good support for such one-way-delay compuation.

* If BBR is in use and ACKs are returned on different paths, it may cause an ambiguity issue with the computation of bandwidth and delay product (BDP). In BBR, BDP is used to limit the number of inflight packets. One may choose to use the smallest RTT measured to compute BDP. However, if the majority of ACKs are returned from a high-latency path, the cwnd = cwnd_gain * bandwidth * min_rtt may be lower than what is needed to achieve good performance. One possible solution is to transmit a new packet and its ACK on the same path. Other possible solutions may include transmitting ACKs on the shortest path with relative increase of cwnd_gain. For the time being, we think there is a research problem and it is up to the implementers to pick the best solution.

16. Appendix.C Difference from past proposals

This proposal differs from past proposals [I-D.deconinck-quic-multipath] in two fundamental perspectives:

* The multi-path QUIC is built on top of the concept of the bidirectional paths, which readily fits into the nature of both cellular and wifi links that cover the majority of multi-path applications in QUIC while keeping the design simple and easy to implement. In doing so, we are able to re-use most of the current QUIC transport design with the sole addition of three new frames.

* The multi-path QUIC design enables feedback-based dynamic scheduling strategy. As the major goal of multi-path QUIC is to enhance performance in mobile applications, where the sender and receiver may have different viewpoints about the fast-changing wireless connectivity, especially in high-mobility scenarios, the proposed design allows the sender and receiver to synchronize their viewpoints via message exchange in ACK packet in order to maximize performance.

17. References

17.1. Normative References

[QUIC-DELAYED-ACK]
Iyengar, J., Ed. and I. Swett, Ed., "Sender Control of Acknowledgement Delays in QUIC", Work in Progress, Internet-Draft, draft-iyengar-quic-delayed-ack-02, <https://tools.ietf.org/html/draft-iyengar-quic-delayed-ack-02>.

[QUIC-LB] Duke, M., Ed. and N. Banks, Ed., "QUIC-LB: Generating Routable QUIC Connection IDs", Work in Progress, Internet-Draft, draft-ietf-quic-load-balancers, <https://tools.ietf.org/html/draft-ietf-quic-load-balancers>.

[QUIC-RECOVERY]
Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", Work in Progress, Internet-Draft, draft-ietf-quic-recovery, <https://tools.ietf.org/html/draft-ietf-quic-recovery>.

[QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-tls, <https://tools.ietf.org/html/draft-ietf-quic-tls>.

[QUIC-TRANSPORT]
Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport, <https://tools.ietf.org/html/draft-ietf-quic-transport>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

17.2. Informative References

[I-D.deconinck-quic-multipath]
Coninck, Q. and O. Bonaventure, "Multipath Extensions for QUIC (MP-QUIC)", Work in Progress, Internet-Draft, draft-deconinck-quic-multipath-06, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-deconinck-quic-multipath-06.txt>.

[MPTCP-LIA]
Raiciu, C., Handly, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", October 2011, <https://tools.ietf.org/html/rfc6356>.

[MPTCP-OLIA]
Khalili, R., Gast, N., and J. Boudec, "Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP", July 2014, <https://datatracker.ietf.org/doc/html/draft-khalili-mptcp-congestion-control-05>.

[QUIC-Timestamp]
Huitema, C., "Quic Timestamps For Measuring One-Way Delays", August 2020, <https://datatracker.ietf.org/doc/draft-huitema-quic-ts/>.

Authors' Addresses

Yanmei Liu
Alibaba Inc.

Email: miaoji.lym@alibaba-inc.com

Yunfei Ma
Alibaba Inc.

Email: yunfei.ma@alibaba-inc.com

Christian Huitema
Private Octopus Inc.

Email: huitema@huitema.net

Qing An
Alibaba Inc.

Email: anqing.aq@alibaba-inc.com

Zhenyu Li
ICT-CAS

Email: zyli@ict.ac.cn