Multipath extension for QUIC

Draft-ietf-quic-multipath-10

QUIC meeting @ IETF-120 Vancouver

Yanmei Liu, Yunfei Ma, Quentin De Coninck,
Olivier Bonaventure, Christian Huitema, Mirja Kühlewind

Agenda

- Brief Summary: Changes from draft -06 to draft -10
- Hackathon Interop Reports: draft-10
- Open issues & Needs-discussion pull requests

Updates of draft -07 / -08

- Merged explicit path ID proposal (draft -07)
- Clarification that the same path ID is used on both ends
- Clarification of the MAX_PATHS frame
- Clarification that after a PATH_ABANDON was sent all CIDs belonging to the path ID have to be retired and only then the path ID can be released/retired

Updates of draft -09 (after interim discussion)

- The transport parameter previously known as 'initial_max_paths' has been renamed to 'initial_max_path_id' (the updated value 0x0f739bbc1b666d09), containing the max path id instead of path count; also change the MAX_PATH_ID frame for the same semantics
- Rewrite the Path Closure section: Endpoints send PATH_ABANDON frames in each scenes as an explicit signal for path closure, and endpoints don’t necessarily need to send MP_RETIRE_CID for CIDs belonging to the closing path.
- Recommendations regarding Path ID continuity
- Editorial Changes: reconstruct of the high level introduction and acknowledgement sending subsection, also rewrite the examples of path initiation and path closure
- First update for Security Consideration Section

Updates of draft -10

- Specify details of MP_RETIRE_CONNECTION_ID
- Rename ACK_MP frame to MP_ACK without semantic change
- Allow multiple paths use the same 4 tuple
- Limit initial_max_path to 2^32-1 and explain more about the limitation
- More guidance on path status "standby"
- Editorial minor change including error handling clarification, normative language for max path id, and stream frame boundaries of retransmissions

Hackathon Interop Reports: draft-10

server / client ↓	xquic	picoquic	Rask	quiche
xquic	HVDCISURA	HVDCISUAR	HVDCISUAR	HVDCISUAR
picoquic				
Rask	HVDCSUARM	HVDCSUARM	HVDCSUAR	HVDCSUAR
quiche	HVDCISAR	HVDCISAR	HVDCISAR	HVDCISAR

Feature	code	details
Handshake	H	The handshake completes with successful negotiation of enable_multipath transport parameter (both ends indicate 0x01)
Path Validation	V	Client sends PATH_CHALLENGE frame to open a new path and server replies with PATH_RESPONSE
Send data	D	Stream data (of one of more streams) is send on all paths; ACK_MP frames are sent and processed
Path Close	C	Client closes a path with PATH_ABANDON frame
Optional Features Tested		
Feature	**code**	**details**
CID change	I	A server offers new CIDs to a client in advance. Upon some events, the client starts using a new server CID on one path
Path status	S	Client sends PATH_STANDBY and PATH_AVAILABLE frames
Key Update	U	One endpoint updates keys and sends at least one packet with the new key on all active paths
Multipath ACK	A	One endpoint sends data and the other endpoints sends ACK (randomly) on all path independent of where data is received
CID retirement	R	One endpoint send an RETIRE_CONNECTION_ID for an active path
Migration	M	Change CID and 4-tuple e.g. port on an existing path

Issue review part 1 -
Mostly editorial issues aka naming frames ;-)

Issue #359: Naming consistency of frames

- Currently, the new frames are
  - MAX_PATH_ID
  - PATH_ABANDON, PATH_AVAILABLE, PATH_STANDBY
  - MP_ACK, MP_NEW_CONNECTION_ID, MP_RETIRE_CONNECTION_ID

- Proposal is to harmonize to
  - PATH_ID_MAX
  - PATH_ABANDON, PATH_AVAILABLE, PATH_STANDBY
  - PATH_ACK, PATH_CONNECTION_ID, PATH_RETIRE_CONNECTION_ID

Issue #283: Sending PATH_STANDBY before the connection ID is used is pointless

- The discussion on the issue indicated that it could be clearer that this frame is a (unidirectional) recommendation but cannot be binding, especially as a standby path maybe used anytime if the active paths seem broken.

- Proposed solution
  - Rename PATH_STANDBY to PATH_BACKUP to make this use case more clear

Issue review part 2 - More design-like issues

Issue #343: What if SPA migration fails? (PR #403)

- Based on RFC 9000: The disable_active_migration transport parameter indicates that "an endpoint ... MUST NOT use a new local address when sending to the address that the peer used during the handshake."
- What's different with multipath? (This is clarified in PR #403)
  - disable_active_migration limits the new path creation using the peer's handshake address
  - But, opening new paths for any local address to other peer addresses e.g the server's preferred address (SPA) can immediately be done
- Reasoning scenarios (same of the RFC 9000): Server-side Anycast IP
  - The migration to the server preferred address (SPA) will allow a direct connection to the server
  - But connecting from a different source address may land on a different destination instance
- Question: Or is there anything else to consider that would limit the opening of new paths before migration to the preferred address?

Issue #220: "SHOULD" use ACK_MP frames?

- Current text (in editorial PR #419 which is not merged yet)

```
Similarly after a successful handshake, endpoints SHOULD also use
the MP_NEW_CONNECTION_ID frame to provide new connection IDs for Path ID 0 and,
respectively, the MP_RETIRE_CONNECTION_ID frame to retire connection IDs for
Path ID 0.
```

- Discussion on MAY, SHOULD, or MUST?
  - MUST: It easier to handle only one kind of Frame
  - MAY: 1) safe one byte on path 0, 2) similar behavior as RFC9000 if only path O is used
- Proposed solution
  - Keep SHOULD (close without action)
  - But should we add more discussion/guidance?

Issue #378: Do we need further guidance if multiple paths over the same 4-tuple are used?

- PR #400 (merged) removes the restriction for only one path per 4-tuple as this is not not necessary with the explicit Path ID approach
- Question: Do we need to provide further guidance?
  - E.g. on closing paths on the same 4-tuple if that is not desired by one peer
    - Specially avoid that both ends close a different paths
  - Use of congestion control or packet/ACK scheduling

Issue #253: Review error codes

- There are currently 4 case where we send an error code:
  1. multipath is negotiated but there is no CID(s) -> MP_PROTOCOL_VIOLATION
  2. Cipher suites with too short nonce -> TRANSPORT_PARAMETER
  3. multipath frame within wrong packet type -> FRAME_ENCODING_ERROR
  4. seq number of CID (used in a frame) was is not valid -> MP_PROTOCOL_VIOLATION
- Question
  - Are these the right error codes? Do we really need a extension-specific error code?
- Possible solution
  - Remove MP_PROTOCOL_VIOLATION and use PROTOCOL_VIOLATION instead
  - And/or: Do we want/need a more specific error code if no CID is used?

Issue #414: Abandon Frame needs Error Code definition

```
PATH_ABANDON Frame {
    Type (i) = TBD-02 (experiments use 0x15228c05),
    Path Identifier (i),
    Error Code (i),
    Reason Phrase Length (i),
    Reason Phrase (..),
}
```

- Proposal
  - Use Application Protocol Error Code (as in STOP_SENDING frame)
  - Use error code = 0 for abandon by QUIC layer (on time-out and in response to an abandon by the peer)

Issue #342: MAX_PATHS_BLOCKED(PR #402)

- Proposal to add new MAX_PATHS_BLOCKED frame to indicate if a peer would need CIDs for a new Path ID
- Currently it is recommended to send a MAX_PATH_ID frame to inform the peer that it could use new active path identifiers
- Question: Should we do this?

Issue #312: PATH_STANDBY PATH_AVAILABLE Sequence number spaces

- Currently there is one sequence number space for PATH_STANDBY and PATH_AVAILABLE for the whole connection
- Alternatively, we could define sequence number spaces per path (Path ID), similarly to MP_(NEW|RETIRE)_CONNECTION_ID frames (PR #396)
- Both works but each case is either slightly more state at the sender or receiver.
- Question: What should we do?

Issue #303: Recommendation on token ambiguity issue ~~is violating RFC 9000~~

- Current text

  ```
  To alleviate such a token ambiguity issue, a server may issue a token
  that is capable of validating any of the previously validated
  addresses.
  ```

- Client doesn’t know for which addresses the token applies, however, in the worst case the address has to be validated again and this can always happen anyway
- Proposal/Question
  - Should we rather require that the token can be used to validate all of the validated addresses of the connection?

Issue #362: Clarify that "refusing a path" means "closing a path" (PR #388)

PR #388 removes a separate section on “Refusing a path” and clarifies that it is recommended to send an PATH_ABANDON frame on another path if a PATH_CHALLANGE is received but the receiver does not want to open a new path.

Issue #397: What to do on path time-out (See also PR #398)

- Current text

  ```
  Hosts SHOULD stop sending traffic on a path if for at least
  the period of the idle timeout.
  ```

  - Note PR #377 (merged in -09) requires to send an PATH_ABANDON also on close after idle timeout (on another active path).
- Question
  - But do we really have to close the path on idle timeout?
  - Using keep-alives is really local decision depending on potentially knowledge of the network
  - Sending keep-alives without a need wastes ressource (e.g. wake-up the radio) but you might still want to use the path later (standby)
- Proposal
  - Use MAY (or no normative language at all) + provide more guidance

Issue #390: What if a packet with a CID belonging to a new Path ID without path challenge is received?

- Current text

  > When the multipath extension is negotiated, a client that wants to use an additional path MUST first initiate the Address Validation procedure with PATH_CHALLENGE and PATH_RESPONSE frames as described in Section 8.2 of [QUIC-TRANSPORT], unless it has previously validated that address.

- Thus receiving a packet without a path challenge on an non-validated path is a protocol violation, however, we don't specify what to do in this case…?
- Solutions/Question
  - 3 options: send path challenge, ignore, or connection error