Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 7 December 2025

T. John
T. Riechard
Otto-von-Guericke University Magdeburg

5 June 2025

Deadline Aware Streams in QUIC Multipath
draft-tjohn-quic-multipath-dmtp-01

Abstract

This document proposes the Deadline-aware Multipath Transport Protocol (DMTP) concept as an extension to the Multipath Extension for QUIC (QUIC-MULTIPATH) as well as QUIC itself. This extension aims to support data streams with strict latency requirements by enabling the signaling of a stream's deadline and ideally by combining multipath scheduling, congestion control adaptations, and optional forward error correction (FEC). Moreover, DMTP leverages the path identifiers introduced by the Multipath Extension for QUIC to distinguish different end-to-end paths as they may be offered in a Path Aware Network (PAN) such as SCION. This allows an application to select its preferred paths while maintaining interoperability with standard endpoints using the Multipath Extension for QUIC. In combination, DMTP enables endpoints to exchange and schedule deadline-aware streams across multiple network paths. Additionally, this proposal also maintains compatibility with QUIC itself, in order to deliver its benefits - albeit with limited effectiveness - even in scenarios where only a single path can or should be used.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://netsys-lab.github.io/mpquic-dmtp-draft/draft-tjohn-quic-multipath-dmtp.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-tjohn-quic-multipath-dmtp/.

Source for this draft and an issue tracker can be found at https://github.com/netsys-lab/mpquic-dmtp-draft.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 7 December 2025.

Copyright Notice

Table of Contents

1. Introduction

The Multipath Extension for QUIC [QUIC-MULTIPATH] enhances performance by simultaneously utilizing multiple paths between endpoints. However, both [QUIC] and [QUIC-MULTIPATH] currently lack support for strict deadline requirements of real-time applications such as teleoperation, live video streaming, or interactive gaming, which are becoming increasingly important. These applications demand low and bounded latency, and can often tolerate no or only partial retransmission of missing data.

Previous and ongoing work on deadline-aware protocols for [QUIC] includes the Deadline-aware Transport Protocol [QUIC-DTP] as well as Media over QUIC Transport [MOQT], both single-path-only approaches that introduce deadlines for streams but do not exploit multipath capabilities. Meanwhile, our [DMTP] approach additionally allows taking advantage of multiple paths, combined with forward error correction (FEC) and intelligent retransmissions, to significantly increase the fraction of packets meeting their deadlines, especially in the presence of lossy or high-latency links.

By integrating deadline-aware concepts into [QUIC-MULTIPATH], we seek to enable:

1. Multipath streams with Deadlines: Scheduling and transmitting data across multiple paths, with strict deadlines of streams driving scheduling decisions.

2. Option for Path-Aware Networking: Support for path selection as offered in Path Aware Networks (e.g., [SCION]) by mapping each potential path to an [QUIC-MULTIPATH] path identifier.

3. Deadline-Based Retransmission / FEC: Combining optional adaptive FEC (such as [QUIC-AFEC]) and "smart" retransmissions only when there may still be time left to meet the deadline.

Using the deadline-aware concepts of this proposal together with [QUIC] in a single-path scenario will still offer the advantage of deadline-based retransmissions / FEC but limit its effectiveness to the boundaries set by the available path.

This draft specifies a minimal set of protocol extensions for [QUIC-MULTIPATH] and [QUIC] respectively to exchange deadline information at the transport layer, so that endpoints can coordinate scheduling for multipath transmissions with strict time constraints. One goal of this proposal is to gather community feedback and gauge interest to guide future refinements.

1.1. Motivation and Applications

Real-time applications often produce data blocks (e.g., video frames or control messages) that are only valuable if delivered before a specific deadline. Example use cases include:

* Teleoperation and Remote Control: Robotic control or telepresence systems require deterministic and low latency feedback. Missed control signals, sensor data or video frame deadlines can lead to system instability or degraded user experience.

* Live Streaming and Interactive Media: Latency-sensitive video or audio streams (e.g., for live concerts, online VR gaming, cloud rendering) benefit from leveraging multiple paths to sustain low-latency delivery even under varying network conditions.

* Online Gaming: Multiplayer networked games exchange frequent, time-critical state updates. Late updates are effectively wasted, so an approach to drop or deprioritize old data can save bandwidth and improve real-time responsiveness.

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.

Within this document:

* "Deadline-aware streams" refers to streams in which an application indicates a time by which data must be delivered, beyond which data is no longer useful.

* "Path" aligns with the [QUIC-MULTIPATH] concept: Each path is identified by a unique Path ID, and it may reference a specific combination of source and destination IP:port tuples or a distinct end-to-end path as they may be offered in a path aware network, as defined by [RFC9473].

* A connection that is "multipath-capable" in the sense of this draft is defined as a connection in which both endpoints negotiate an initial_max_path_id transport parameter from [QUIC-MULTIPATH] to a value greater than 0.

* A connection that is "multipath-active" in the sense of this draft is defined as a connection that currently has two or more validated paths. Packets may be transmitted on any of these paths.

3. Design Overview

3.1. Integrating Deadline-Aware Streams into QUIC-MULTIPATH/QUIC

Our design goal is to extend [QUIC-MULTIPATH] and [QUIC] respectively with minimal changes. The proposed extension enables endpoints to signal a stream’s deadline. Implementations that support deadline-aware streams MUST support:

1. Packet Scheduling: Implement scheduling algorithms that:

    * Prioritize packets from streams with earlier deadlines

    * Account for path characteristics when making scheduling decisions

    * Consider current congestion state of available paths

2. Retransmission Control: Implement retransmission policies that:

    * Evaluate whether retransmitted packets can meet remaining deadlines

    * Skip retransmissions when deadlines cannot be met

    * Consider path conditions when selecting retransmission paths for a multipath-capable connection

3. Deadline Monitoring: Track deadline status and:

    * Detect when deadlines cannot be met

* Signal deadline misses to the application layer

* Allow applications to specify handling of missed deadlines

Additionally, Implementations supporting deadline-aware streams with multipath-capable connections MUST feature:

1. Path Selection: Select paths for transmitting frames, retransmissions and acknowledgements based on metrics relevant to meeting deadlines, including:

    * Path latency measurements

    * Estimation of available bandwidth

    * Observed packet loss rates

Implementations MAY also choose to support:

1. Optional Forward Error Correction: MAY implement FEC mechanisms that:

    * Reduce retransmission overhead for deadline-sensitive streams

    * Adapt FEC overhead based on path conditions

    * Apply FEC selectively based on stream requirements (e.g., stream priority)

3.1.1. Extensions to QUIC-MULTIPATH/QUIC

Our extensions build on [QUIC-MULTIPATH]’s multipath framework (e.g., paths, path IDs, and validation). It will, however, also work with connections that are not multipath-capable, be it with a reduced feature set (see Section 3.1). This extension will add only:

* A transport parameter to enable deadline-aware streams.

* A DEADLINE_CONTROL frame to signal stream deadlines.

* Optional support for an ACK_EXTENDED frame for improved per-path delay feedback.

* Optional AFEC support via an extra transport parameter and two frames for source and repair symbol metadata.

This preserves the original wire format, ensuring interoperability with both [QUIC-MULTIPATH] and [QUIC] endpoints that do not implement these extensions.

3.2. Support for Path-Aware Networks (PAN)

When operating over a path-aware network in addition to [QUIC-MULTIPATH], endpoints can discover and utilize multiple disjoint or partially disjoint paths. This can be provided, for example, by [SCION] or other architectures such as [SR]. In such an environment, a single source-destination address pair may yield multiple distinct end-to-end paths, each with unique performance characteristics (e.g., latency, loss rate). These paths can be exposed to the transport layer via distinct Path IDs in [QUIC-MULTIPATH].

This document does not prescribe how endpoints discover and enumerate available paths at the network layer. Rather, it assumes that a PAN can supply multiple viable routes between endpoints. Once discovered, each route is mapped to a unique Path ID, enabling the [DMTP] scheduling logic to treat them as distinct transport paths for deadline-aware streams.

Multiple paths provided by [QUIC-MULTIPATH] on the one hand and path diversity provided by PAN on the other hand enhance the effectiveness of [DMTP]’s scheduling, retransmission, and optional FEC mechanisms in meeting strict deadlines, making support for PAN essential to the design of the proposed extension.

3.3. Deadlines

3.3.1. Signalling Deadlines

To signal deadlines, endpoints use the DEADLINE_CONTROL frame (see Section 4.2). This frame associates a specific deadline with a stream, indicating the relative time by when the data should be delivered.

3.3.2. Deadline Semantics

* Deadline Representation: Deadlines are represented as a relative time in milliseconds from the time the frame is sent.

* Stream Association: A deadline applies to a specific stream identified by its Stream ID

* Transport Behavior: Upon receiving a DEADLINE_CONTROL frame, the transport layer SHOULD attempt to schedule and retransmit packets carrying data for the specified stream to meet the indicated deadline.

* Retransmissions and Scheduling: Endpoints MAY implement custom schedulers and congestion controllers optimized for deadline-aware traffic, such as those based on [DMTP] concepts.

3.3.3. Handling Missed Deadline

If the transport layer determines that the deadline cannot be met, it MAY choose to:

* Discard the data associated with the deadline-aware stream.

* Inform the application of the missed deadline.

* Continue delivering the data if it is still deemed useful.

The specific behavior is implementation-specific and MAY be configurable by the application.

3.4. Adaptive Forward Error Correction (FEC)

When deadlines are tight and packet losses frequent, relying solely on retransmissions may cause data to miss its deadline. To mitigate this risk, this extension optionally uses Adaptive FEC (AFEC) as proposed in [QUIC-AFEC]. AFEC can reduce the need for retransmissions, particularly in networks with random or bursty loss characteristics.

When using AFEC in a multipath-capable connection, the Tag Type of the FEC_Tag MUST be set to 1 to indicate "Long Flow Usage". In turn, both source symbol packets and repair symbol packets MUST carry the FEC_Tag frame so that repair packets can be correctly matched to their corresponding source packets across different paths. Such a constraint is not needed in a connection that is not multipath-capable.

In a multipath-active connection, FEC repair packets SHOULD be sent over a path different from the one carrying the source data. This de-correlates losses and increases the likelihood that repair symbols arrive even if other paths experience congestion or packet loss. The coding rate (i.e., the ratio of repair symbols to source symbols) MAY be configured on a per-stream basis, depending on the stream’s tolerance for overhead versus its deadline sensitivity.

3.5. Smart Retransmissions

Smart retransmissions in a deadline-aware context mean that lost frames are only retransmitted if there is still enough time left to meet the deadline via one or - with a multipath-active connection - more paths. The sender computes whether the frames can arrive on time, factoring in the path's estimated one-way delay or RTT. If not, the sender discards the frames rather than wasting congestion window or scheduling capacity.

3.6. Path Metrics

To schedule traffic effectively, the sender SHOULD gather:

* One-Way Delays or RTT for determining if the data can reach its destination before the deadline and in case of a multipath-capable connection for selecting the path(s) that can deliver data before the deadline.

* Loss Rate: For deciding whether to apply adaptive FEC or more aggressive retransmissions.

* Available Bandwidth: So that sending on path(s) with insufficient capacity does not cause additional delay.

3.6.1. Per Path Delay

A crucial metric for DMTP is the one-way or round-trip delay of the available path(s). This is used to decide whether a new or retransmitted packet can arrive before its deadline. In a path-aware network, the one-way delay might be advertised or inferred from routing information. Otherwise, endpoints measure RTT or one-way delay themselves.

For accurate one-way delay measurements, endpoints MAY use synchronized clocks; if full clock sync is not feasible, a fallback to round-trip time measurements is still acceptable. For improved delay tracking, the additional fields for the receive timestamp of the ACK_EXTENDED frame as proposed in [QUIC-RECEIVE-TS] is used.

With a multipath-capable connection, if endpoints have agreed on the usage of the ACK_EXTENDED frame with the additional receive timestamp fields (Bit 1 of extended_ack_features transport parameter), packets containing a PING (type=0x01) frame MUST be acknowledged on the same path that the packet was received on.

3.6.2. Gathering Path Metrics

1. Path-Aware Networks might provide direct metrics, such as path latency or bandwidth as part of path metadata.

2. Active Probing: If the underlying network does not provide metrics, the endpoint MAY send periodic PING frames or small test packets along each active path.

3. Path Measurement Frames: This draft uses the ACK_EXTENDED frame ([QUIC-RECEIVE-TS]) for deeper path measurements, including timestamps of packet receipts to estimate per path one-way delay.

4. Congestion windows, RTT estimates, and packet loss detection from [QUIC-MULTIPATH]’s standard loss recovery can inform scheduling.

4. Extension to QUIC-MULTIPATH/QUIC

This extension builds upon [QUIC-MULTIPATH] and [QUIC] respectively. Below we list the protocol additions and modifications. Unless otherwise noted, all rules of [QUIC-MULTIPATH] or [QUIC] remain.

4.1. Handshake Negotiation and Transport Parameter

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

* enable_deadline_aware_streams (value TBD): A zero-length value that, if present, indicates that the endpoint supports deadline-aware streams.

Endpoints negotiate the use of deadline-aware streams by including the enable_deadline_aware_streams transport parameter in the handshake. Both endpoints MUST include this transport parameter to enable the use of deadline-aware streams. If an endpoint receives a DEADLINE_CONTROL frame without having negotiated support, it MUST treat this as a connection error of type PROTOCOL_VIOLATION

4.2. DEADLINE_CONTROL Frame

The DEADLINE_CONTROL frame (type=TBD) is used to signal deadline-awareness for specific streams and to indicate their associated deadlines.

```
DEADLINE_CONTROL Frame {
  Type (i) = TBD,
  Stream ID (i),
  Deadline (i),
}
```

The DEADLINE_CONTROL frame contains the following fields:

Stream ID: A variable-length integer indicating the Stream ID to which the deadline applies.

Deadline: A variable-length integer representing the relative deadline in milliseconds from the time the frame is sent. An endpoint sends a DEADLINE_CONTROL frame to indicate that data on the specified stream should be delivered by the given deadline. Upon receiving this frame, the peer MUST attempt to schedule and deliver the data on the specified stream within the indicated deadline.

Usage Constraints:

* Endpoints MUST NOT send the DEADLINE_CONTROL frame unless both endpoints have negotiated support via the enable_deadline_aware_streams transport parameter.

* If an endpoint receives a DEADLINE_CONTROL frame without having negotiated support, it MUST treat it as a connection error of type PROTOCOL_VIOLATION.

* The DEADLINE_CONTROL frame MUST only be sent in 1-RTT packets.

5. API

Though this draft primarily focuses on wire-level protocol changes, an implementation that exposes a user-level API might provide:

* GetDeadlineAwareStreams(connection): Returns a set of tuples for the given connection (see Section 5.1) in the form of (stream_id, deadline_ms) where a value of 0 for the deadline_ms indicates, that endpoints have agreed on using deadline-aware streams but no DEADLINE_CONTROL Frame has been sent (yet)

* SetStreamDeadline(connection, stream_id, deadline_ms): Informs the transport that data on stream_id must arrive before deadline_ms.

* GetAvailablePaths(connection): (Optional for use with PAN) Retrieves the available paths between the endpoints from the underlying PAN and returns them as a set of strings, each representing one path. The representation of a path is dependent on the used PAN.

* SetPaths(connection, pan_paths): (Optional for use with PAN) Defines the subset of available paths (a set of strings) to be used. This applies to all the streams inside the connection. Depending on the underlying PAN, the string(s) might include wildcards or other operators that can be interpreted by the PAN.

* GetStreamPaths(connection, stream_id): Returns a set of tuples in the form of (path_id, role) where role describes the role of the path in the stream. Possible values are:

  - data: The path(s) over which data is transmitted

  - retransmission: The path that is used for retransmissions and acknowledgements

  - backup: Path(s) that can be used if any data path(s) become(s) unavailable

  - none: Path(s) that will not be used

* SetStreamPaths(connection, stream_id, paths): (Optional for use with external optimizer) Allows defining, which paths should be used for a given stream_id. paths is a set of tuples in the form of (path_id, role, fraction). Available paths that are omitted here will receive the role none. fraction may only be used on paths with the role data and is only effective if there is more than one data path.

* GetPathMetrics(connection, stream_id, path_id): Returns a set of key-value pairs (KVP) which characterize the path. Possible KVPs are:

  - role: Describes the role of the path in the connection; Possible values: data, retransmission, backup, none

  - bandwidth: the bandwidth in bits/s of the path as measured or as signaled by the PAN.

  - owd: One way delay in ms of the path as measured or as signaled by the PAN; MUST only be populated, if rtt is left empty.

  - rtt: Round trip time in ms of the path as measured or as signaled by the PAN; MUST only be populated, if owd is left empty.

  - loss_rate: Loss rate of the path as measured.

  - costs: If a metric for the cost of a path is available, it may be included here

* OnMissedDeadline(connection, stream_id): (Optional) callback that the transport can invoke if data is considered impossible to deliver on time. The application can choose to send new data, discard, or do nothing.

These calls let an application specify deadlines and priorities dynamically.

5.1. Commonly Used Data Structures

* Connection: The connection is the 4-tuple introduced by [QUIC] which identifies a connection. It is structured like (source_IP, source_port, destination_IP, destination_port)

6. Security Considerations

This extension retains all the security features and considerations of [QUIC], [QUIC-TLS] and [QUIC-MULTIPATH]. Nevertheless, it introduces additional considerations:

* Deadline Signaling: Knowledge of deadlines or priorities may be sensitive if it reveals application timing patterns or critical data intervals. Implementations SHOULD carefully handle metadata (e.g., by encrypting frames in 1-RTT packets).

* Resource Exhaustion and Flooding: The ability to manage multiple concurrent paths and to schedule or drop data based on deadlines must not weaken [QUIC]’s anti-amplification measures. Endpoints MUST still follow [QUIC] path validation procedures, ensuring that an attacker cannot exploit deadline-aware frames to amplify traffic.

* When employing ACK_EXTENDED frames for one-way delay measurement with clock synchronization, the clock synchronization must also be secured. Otherwise, an attacker injecting false timestamps could mislead scheduling. Endpoints that rely heavily on these measurements should be aware of that risk and possibly cross-check with measured RTT or other heuristics.

7. IANA Considerations

This document defines a new transport parameter for the negotiation of deadline-aware streams for [QUIC-MULTIPATH] or [QUIC], and one new frame type. The draft defines provisional values for experiments.

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	enable_deadline_aware_streams	Section 4.1

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 (1)	DEADLINE_CONTROL	Section 4.2

Table 2: Addition to QUIC Frame Types Entries

8. References

8.1. Normative References

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[QUIC-AFEC] Moskvitin, D., Onegin, E., Huang, R., Luo, H., and Q. Chen, "Adaptive Forward Erasure Correction for Delay-Sensitive QUIC Connections", Work in Progress, Internet-Draft, draft-dmoskvitin-quic-adaptive-fec-00, 6 May 2024, <https://datatracker.ietf.org/doc/html/draft-dmoskvitin-quic-adaptive-fec-00>.

[QUIC-MULTIPATH] Liu, Y., Ma, Y., De Coninck, Q., Bonaventure, O., Huitema, C., and M. K端hlewind, "Multipath Extension for QUIC", Work

in Progress, Internet-Draft, draft-ietf-quic-multipath-14, 23 April 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-14>.

[QUIC-RECEIVE-TS] Smith, C., Swett, I., Beshay, J., Jaiswal, S., Purushothaman, I., and B. Schlinker, "QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps", Work in Progress, Internet-Draft, draft-smith-quic-receive-ts-02, 22 April 2025, <https://datatracker.ietf.org/doc/html/draft-smith-quic-receive-ts-02>.

[QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[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/rfc/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/rfc/rfc8174>.

[RFC9473] Enghardt, R. and C. Kr辰henb端hl, "A Vocabulary of Path Properties", RFC 9473, DOI 10.17487/RFC9473, September 2023, <https://www.rfc-editor.org/rfc/rfc9473>.

8.2. Informative References

[DMTP] John, T., Perrig, A., and D. Hausheer, "DMTP: Deadline-aware Multipath Transport Protocol", IEEE, 2023 IFIP Networking Conference (IFIP Networking), DOI 10.23919/ifipnetworking57963.2023.10186417, June 2023, <https://doi.org/10.23919/ifipnetworking57963.2023.10186417>.

[MOQT] Nandakumar, S., Vasiliev, V., Swett, I., and A. Frindell, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-11, 28 April 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-11>.

[QUIC-DTP] Cui, Y., Ma, C., Shi, H., Zheng, K., and W. Wang, "Deadline-aware Transport Protocol", Work in Progress, Internet-Draft, draft-shi-quic-dtp-10, 29 July 2024, <https://datatracker.ietf.org/doc/html/draft-shi-quic-dtp-10>.

[SCION] de Kater, C., Rustignoli, N., and S. Hitz, "SCION Control Plane", Work in Progress, Internet-Draft, draft-dekater-scion-controlplane-07, 24 December 2024, <https://datatracker.ietf.org/doc/html/draft-dekater-scion-controlplane-07>.

[SR] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/rfc/rfc8402>.

Acknowledgements

The authors thank the QUIC working group and the designers of [QUIC], [QUIC-MULTIPATH] and [QUIC-DTP] for paving the way for deadline-aware features in QUIC. The concept of scheduling data with deadlines over multiple paths builds on numerous discussions about partial reliability, adaptive FEC, and optimal path selection.

Authors' Addresses

Tony John
Otto-von-Guericke University Magdeburg
Email: tony.john@ovgu.de

Till-Frederik Riechard
Otto-von-Guericke University Magdeburg
Email: riechard@ovgu.de