Workgroup:	QUIC
Internet-Draft:	draft-navarre-quic-flexicast-02
Published:	28 February 2026
Intended Status:	Experimental
Expires:	1 September 2026
Authors:	L. Navarre	O. Bonaventure
---	---	---
	UCLouvain	*UCLouvain & WELRI*

Flexicast QUIC: combining unicast and multicast in a single QUIC connection

Abstract

This document proposes Flexicast QUIC, a simple extension to Multipath QUIC that enables a source to send the same information to a set of receivers using a combination of unicast paths and IP multicast distribution trees.

The RFC Editor will remove this note

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-navarre-quic-flexicast/.

Discussion of this document takes place on the QUIC Working Group mailing list (mailto:quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/. Subscribe at https://www.ietf.org/mailman/listinfo/quic/.

Source for this draft and an issue tracker can be found at https://github.com/louisna/draft-navarre-quic-flexicast.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 September 2026.

Copyright Notice

Table of Contents

1. Introduction

Starting from the initial efforts of Steve Deering [RFC1112], the IETF has developed a range of IP multicast solutions that enable the efficient transmission of a single packet to a group of receivers. In this document, we focus on Source-Specific Multicast for IP [RFC4607], but the solution proposed could also be applied to other forms of IP Multicast.

Although IP Multicast is not a new solution, it is not as widely used by applications as popular transport protocols like TCP [RFC9293] or QUIC [QUIC-TRANSPORT] do not support multicast. Current IP Multicast applications include IP TV distribution in ISP networks and trading services for financial institutions. Many reasons explain the difficulty of deploying IP Multicast [DIOT]. From the application's viewpoint, a key challenge with IP Multicast is that even if there exists a unicast path between two hosts, there is no guarantee that it will be possible to create and

maintain a multicast tree between these two hosts to efficiently exchange data using IP multicast. To cope with this problem, applications must implement both a multicast solution and a unicast solution. This increases the complexity and the cost of these applications. For this reason, many applications that send the same information to a large set of receivers, such as streaming services or software updates, still rely on TCP or QUIC over unicast. This is inefficient from the network viewpoint as the network carries the same information multiple times.

The deployment of QUIC opens an interesting opportunity to reconsider the utilization of IP Multicast. As QUIC runs above UDP, it could easily use IP multicast to deliver information along multicast trees, while offering the native reliability and security features of the protocol. Multicast extensions to QUIC have already been proposed in [I-D.pardue-quic-http-mcast] and [I-D.jholland-quic-multicast-08]. To our knowledge, these extensions have not been fully implemented and deployed. Additionally, these solutions suggest to share a QUIC connection between multiple receivers from a specific source, as well as individual connections. This design requires applications to handle two distinct connections in case of packet losses and multicast failures.

Flexicast QUIC takes a different approach. Instead of extending QUIC [QUIC-TRANSPORT], Flexicast QUIC extends Multipath QUIC [MULTIPATH-QUIC]. Multipath QUIC already includes several features that are very useful to allow a QUIC connection to use unicast and multicast simultaneously.

Flexicast QUIC proposes a simple extension to Multipath QUIC that enables to share an additional path between multiple receivers. The destination address of this path can be a multicast IP address and rely on a multicast forwarding mechanism (e.g., IP Multicast) to transmit the packets. This document defines the core design of Flexicast QUIC starting from Multipath QUIC. Side documents will further expand this design to add new features.

This document is organized as follows. After having specified some conventions in Section 2, we provide a brief overview of Flexicast QUIC in Section 3. We describe in more details the Flexicast QUIC handshake in Section 4 and then the new QUIC frames in Section 11.

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.

3. Flexicast QUIC

Multipath QUIC was designed to enable the simultaneous utilization of multiple paths inside one QUIC connection. A Multipath QUIC connection starts with a regular QUIC handshake, and adds the initial_max_path_id transport parameter to negotiate the multipath extension. This parameter specifies the maximum path identifier that an endpoint is willing to maintain at

connection establishment. Multipath QUIC requires the utilization of non-zero connection identifiers and one packet number space per path. Multipath QUIC assumes that the additional paths are created by the client using the server address of the initial path.

Consider as an example a smartphone that uses Multipath QUIC. This smartphone has a Wi-Fi and a cellular interface. The smartphone has created the QUIC connection over the cellular interface. After the handshake, the smartphone and the server have exchanged additional connection identifiers. To create a new path over the Wi-Fi interface, the smartphone sends a PATH_CHALLENGE over this path using one of the connection identifiers advertised by the server. The server replies with a PATH_RESPONSE using one of the connection identifiers advertised by the smartphone. The smartphone confirms the establishment of the second path using a PATH_RESPONSE. At this point that smartphone and the server have two paths to exchange data: the first over the cellular interface and the second over the Wi-Fi interface. Each path uses a different sequence number space, but QUIC packets are encrypted using the same connection keys using a per-path IV construct. When a QUIC packet needs to be sent, the packet scheduler selects the relevant path. If a QUIC frame is lost over one path, it can be retransmitted over any other path.

Flexicast QUIC extends Multipath QUIC by using different encryption and decryption keys. As such, multiple clients can share the same additional path which is encrypted with a different key than their respective unicast path. The IP destination address of this new, shared path **MAY** be a multicast address, so that all receivers attached to this multicast tree receive the same packet.

The case where the IP destination address is *not* an IP multicast address is discussed later in this document. In this case, Flexicast QUIC uses packet replication at the source, e.g., using sendmmsg, using the unicast destination address of the receivers. This solution scales at the source because Flexicast QUIC generates and encrypts only one packet, and duplicating the bytes is straightforward.

A Flexicast QUIC connection starts with a unicast handshake, similarly to a regular QUIC connection. Receivers R1, R2 and R3 establish a connection to source S using the flexicast_support and the initial_max_path_id transport parameters. The initial path between each receiver and the source are individually secured using the keys derived at connection establishement. It remains open during the whole flexicast session and constitute a unicast bidirectional path between the source and each receiver. This path is used for two very different purposes. First, it acts as a control channel and enables the source to exchange control information with each receiver. Second, it can be used to transmit or retransmit data that cannot be delivered along the multicast tree.

Most of the data sent over a Flexicast QUIC connection is transmitted along what we call a multicast flow. A multicast flow is a unidirectional multipath path from a source to a group of receivers. An important characteristic of a multicast flow is that all the packets sent by the source over this flow are secured using keys that are shared between the source and the receivers attached to the multicast flow.

The source advertises the existence of a multicast flow through the FC_ANNOUNCE frame, which is sent on the unicast path with the receiver. This frame contains flow-specific information, such as the destination UDP port and IP address used by the source to send the packets, and the Flexicast Flow ID, i.e., the equivalent of the Connection ID for the multicast flow. When a receiver joins a specific multicast flow, the source additionally sends an FC_KEY frame, which contains the secret required to derive the set of TLS keys used to decrypt packets of the multicast flow.

Flexicast QUIC supports two types of multicast flows. The first type multicast flow is an IP multicast tree, as illustrated in Figure 1. The source relies on an underlying IP multicast network to create an IP multicast tree and selects a set of encryption and authentication keys. The FC_ANNOUNCE frame contains the IP address of the multicast group the source uses to send the data to the receivers. The Flexicast QUIC source uses this multicast flow to efficiently send data to multiple receivers simultaneously (R1 and R2 in Figure 1). R1 and R2 use their unicast path, created at connection establishment, to return acknowledgments. A source can retransmit lost data either on the multicast flow or over a specific unicast path, e.g. when some data was missing at only one receiver.

Flexicast QUIC supports non-multicast-capable receivers in two different ways. A first approach is to use the unicast path to deliver the data to such receivers. This implies that each data must be authenticated and encrypted using the TLS keys derived over each unicast path. This is illustrated in Figure 1 where receiver R3 lies in a non-multicast-capable network and relies on unicast delivery.

This is not efficient when there are multiple receivers. Flexicast QUIC supports a second type of multicast flow which improves performance when there are multiple unicast receivers. For these receivers, it is more efficient from the source viewpoint to encrypt and authenticate a QUIC packet using shared keys and then send a copy of this packet to all receivers, e.g. using system calls such as sendmmsg. In this case, the source maintains a multicast flow where it replicates each packet towards each receiver. It also sends an FC_ANNOUNCE frame to advertise the multicast flow, using the receiver's IP address as a destination instead of a multicast address.

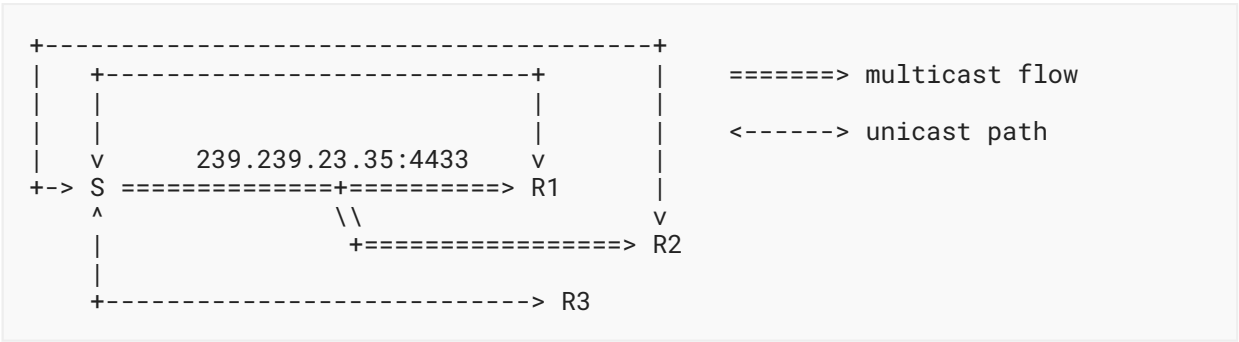

Figure 1: A Flexicast QUIC connection is composed of two types of paths: (1) one bidirectional, unique, unicast path between the source and each receiver and (2) a multicast flow from the source to a set of receivers relying on a multicast distribution tree

Transitions between sending application data on a multicast flow and a unicast path is seamless thanks to the underlying Multipath QUIC extension. At any point in time, if the network conditions of a receiver of a multicast flow degrade, e.g., because the underlying multicast tree fails or because the receiver moves into a non-multicast network, it can leave the multicast flow and continue receiving content through its unicast path.

Flexicast QUIC offers full reliability by retransmitting lost frames either to all receivers on the multicast flow, or per-receiver using their unicast path.

3.1. Extensions to Multipath QUIC.

Flexicast QUIC modifies Multipath QUIC in three ways to enable the utilization of a shared unidirectional path, the multicast flow, as one of the Multipath QUIC paths. The multicast flow can be transported on top of an IP multicast distribution tree to reach several receivers simultaneously, which brings several challenges:

1. An IP multicast tree is unidirectional, in contrast with a Multipath QUIC path.
2. An IP multicast tree is identified by a pair <source address, group address>.
3. All the receivers attached to an IP Multicast tree receive the same packet.

Since an IP multicast tree is unidirectional, it is impossible to use the QUIC path challenge/response mechanism to create the multicast flow along an IP multicast tree. Flexicast QUIC only uses path challenge/response on unicast paths. Flexicast QUIC replaces the explicit path challenge/response from Multipath QUIC to create the multicast flow by an implicit path creation. Since the destination address of this new "path" is a multicast IP address, it is impossible for receivers to test its reachability from the source. An underlying mechanism (e.g., using IGMP) **SHOULD** provide feedback to the receiver on the reachability of the multicast flow through the IP multicast address provided by the source.

Furthermore, the data received over the multicast flow cannot be acknowledged over this path since it is unidirectional. Because Flexicast QUIC uses Multipath QUIC, ACK frames are replaced by PATH_ACK frames. Flexicast QUIC receivers **MUST** send acknowledgments to the source using PATH_ACK frames sent on their unicast path.

In single-source multicast, an IP multicast group is represented by the (S,G) tuple, respectively denoting the IP address of the multicast source and the multicast group. To join the multicast flow, the receivers must be informed of the (S,G) tuple of the underlying IP multicast tree. The server uses the FC_ANNOUNCE frame to advertise the identifier of the multicast tree that the receivers can join.

The Flexicast QUIC packets that are sent over the multicast flow are encrypted and authenticated. However, these packets are not encrypted using the TLS keys derived during the QUIC connection establishment since they are unique to the receiver. To secure the transmission of Flexicast QUIC packets along the multicast flow, the source generates an independent set of security keys and shares them using the FC_KEY frame (see Section 11.3) to all receivers over the

unicast path. Since this frame is sent over the unicast paths, it is authenticated and encrypted using the TLS keys associated to each unicast path. The security questions related to the delivery of the TLS key secrets through the FC_KEY frame are discussed in Section 13.1.

Section 10 gives more details on the modifications to [MULTIPATH-QUIC] to protect packets sent on the multicast flow.

4. Handshake Negotiation and Transport parameter

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

- flexicast_support (current version uses 0xedf3): this transport parameter indicates support of the flexicast extension. The transport parameter contains two boolean values, respectively indicating support of IPv4 and IPv6 for multicast addresses. If an endpoint receives the flexicast_support transport parameter with both IPv4 and IPv6 supports set to false (0), it must close the connection with an error type of FC_PROTOCOL_VIOLATION. The support of the flexicast extension is conditioned to the support of the multipath extension, as defined in [MULTIPATH-QUIC]. As a result, endpoint wishing to support the Flexicast extension **MUST** support [MULTIPATH-QUIC].

An endpoint receiving the flexicast_support transport parameter from its peer, without support for multipath **MUST** ignore the flexicast_support transport parameter.

The extension does not change the definition of the transport parameters defined in Section 18.2 of [QUIC-TRANSPORT].

5. Initialisation of a Flexicast Flow

This section details how a Flexicast QUIC source advertises multicast flows and how receivers join them.

Figure 2 illustrates how a Flexicast QUIC source and receiver exchange frames on the unicast path to advertise and join a multicast flow. The handshake uses the transport parameters defined in the previous section. This handshake creates a QUIC connection between a source and a receiver. The source sends an FC_ANNOUNCE frame over this connection to advertises the multicast flow, e.g. an IP multicast tree. The receiver joins the tree and then sends an FC_STATE(JOIN) frame to the source to indicate that it is attached to the tree. The source sends an FC_KEY frame containing the security keys associated to the multicast flow. The receiver returns an FC_STATE(LISTEN) frame to indicate that it receives frames over the multicast flow.

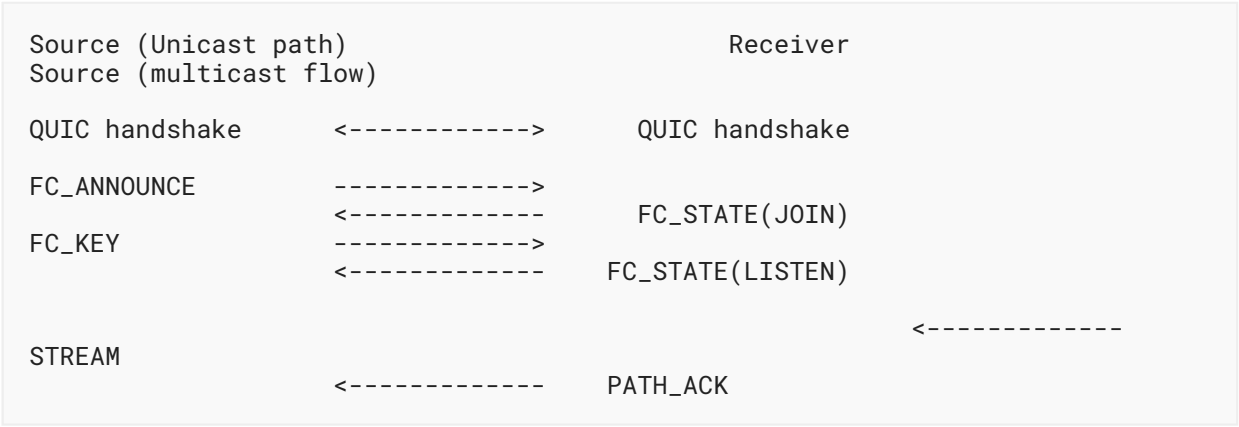

Figure 2: Multicast flow announcement and join

5.1. Flexicast Flow Announcement

A Flexicast QUIC source announces a multicast flow using the FC_ANNOUNCE frame (see Section 11.1). The initial Connection ID derived during connection establishment between the source and the receiver cannot be reused on the multicast flow since it is unique for each receiver. On a multicast flow, Flexicast QUIC replaces the concept of Connection ID with the Flexicast Flow ID. Similarly to the Connection ID from [MULTIPATH-QUIC], the Flexicast Flow ID uniquely identifies a multicast flow. The Flexicast Flow ID **MUST NOT** be empty, as of [MULTIPATH-QUIC] specification. Using a Flexicast Flow ID, a multicast flow can change the addressing at lower protocol layers (UDP, IP) without causing packets to be delivered to the wrong endpoint. In [QUIC-TRANSPORT] and [MULTIPATH-QUIC], endhosts chose their source Connection IDs and advertise them to their peers using NEW_CONNECTION_ID frames. In Flexicast QUIC, the source decides the Flexicast Flow ID identifying a multicast flow and sends this value to receivers.

Through the FC_ANNOUNCE frame, the source advertises the Flexicast Flow ID of the multicast flow to listen to. The advertised Flexicast Flow ID serves as the Destination Connection ID for packets sent on the multicast flow. _TBD: how to avoid colission.

The FC_ANNOUNCE frame also contains the source address, destination address and UDP destination port number of the multicast flow. If the destination address is an IP multicast address, then the multicast flow uses an IP multicast tree and the receiver **MUST** join this tree and listen to the specified UDP port number. If the destination address is the unicast IP address of the receiver, then the receiver **MUST** listen to the specified UDP port number on this address. _TBD: how to avoid colission with an already bound UDP destination port.

The source **MAY** advertise multiple distinct multicast flows to a given receiver, i.e., multicast flows with distinct Flexicast Flow ID. The source **MAY** advertise updated information about a specific multicast flow by sending new FC_ANNOUNCE frames with increased sequence numbers. Upon reception of a new FC_ANNOUNCE frame updating information of an existing multicast flow with a smaller sequence number compared to the last received FC_ANNOUNCE frame, a receiver **MUST** silently discard the new FC_ANNOUNCE frame.

The source **MAY** withdraw a multicast flow by sending a new FC_ANNOUNCE frame, with an increased sequence number, with null source and destination IP addresses. Upon reception of such frame, receivers **MUST** stop listening to packets received from the multicast flow.

5.2. Joining a Flexicast Flow

Figure 3 illustrates the finite-state machine of a receiver from the start of the QUIC connection (Unaware) until it is ready to listen to packets on the multicast flow. After receiving an FC_ANNOUNCE frame advertising a multicast flow, a receiver **MAY** decide to join it. For example, the source can advertise multiple flows disseminating the same video stream in different qualities, letting the receiver choose the appropriate quality. Due to the scalability benefits from the point of view of the source and the network, it is encouraged that receivers do join the multicast flow. Multicast flow management is handled with the FC_STATE frame (see Section 11.2). The receiver sends an FC_STATE frame with the Flexicast Flow ID of the multicast flow it wants to join and the JOIN action.

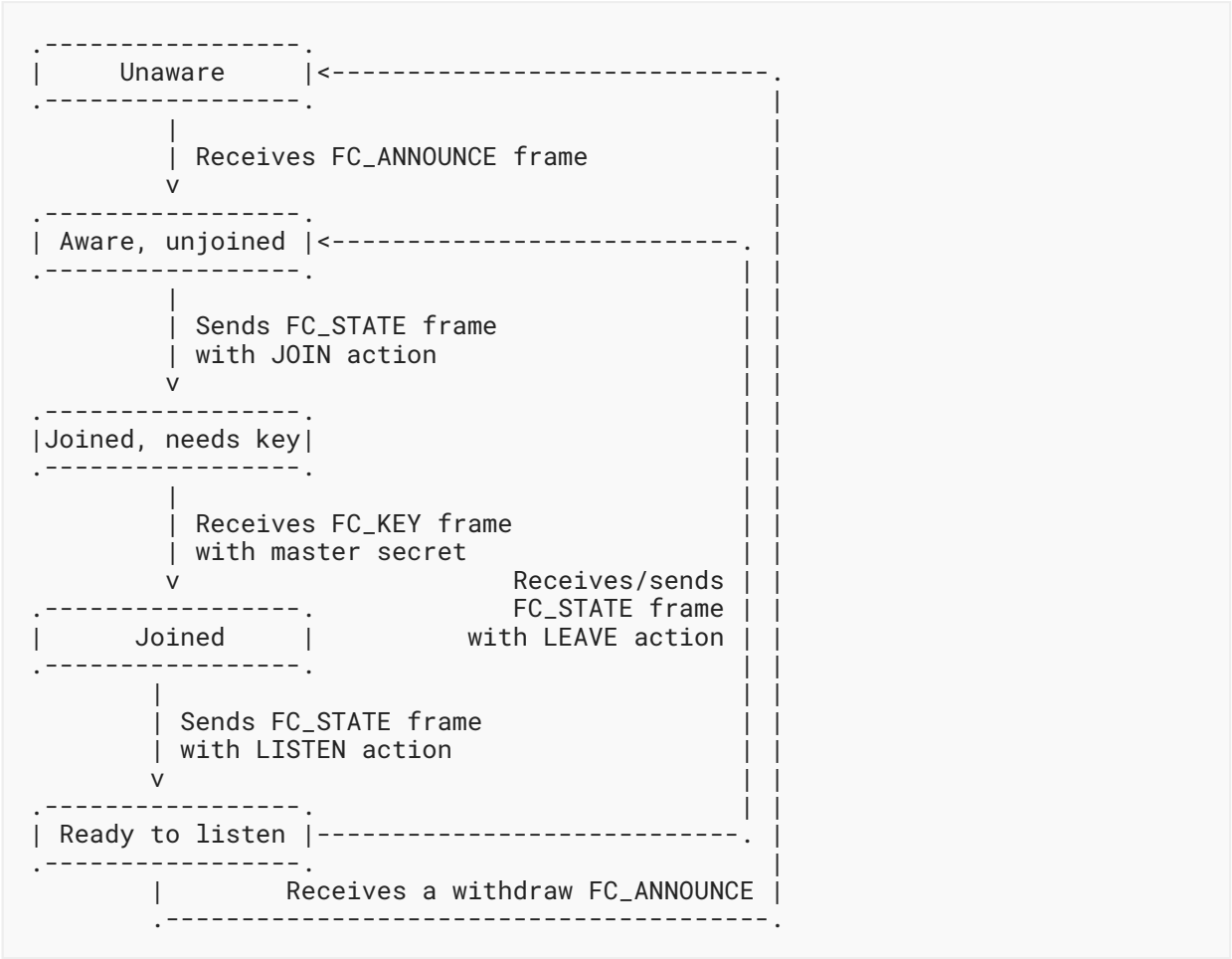

Figure 3: Receiver-side finite-state machine for a given multicast flow ID

Upon reception of an FC_STATE frame with the JOIN action, the source sends an FC_KEY frame, containing the TLS master secret that will be used to derive the set of keys necessary for the receiver to decrypt packets received on the multicast flow. The frame contains an indication on the current packet number sequence such that the receiver can correctly reconstitute the full packet number when receiving a QUIC packet on the flexicast flow The receiver **SHOULD NOT** process packets with a lower packet number, and applications **SHOULD** transmit through the unicast path any application data required for the receiver to correctly process subsequent packets received on the multicast flow ID. Starting from the advertised first packet number of interest, the source **MUST** retransmit on any path any lost reliable frames.

Once the receiver received the FC_KEY frame and its underlying multicast network is ready to receive packets on the multicast flow, it sends an FC_STATE frame to the source with the LISTEN action.

The Flexicast QUIC source **SHOULD NOT** consider the receiver as an active member of the multicast flow before it has received an FC_STATE with the LISTEN action from the receiver. This process avoids the unlikely case where the receiver wants to join the multicast flow, but no underlying multicast distribution tree could be constructed toward the receiver (e.g., it lies in a unicast-only network). By not considering such receiver as an active member of the group, the Flexicast QUIC source avoids considering all sent packets as lost, potentially impacting its congestion state.

5.3. Underlying (multicast) network

If the IP destination address in the FC_ANNOUNCE frame is a multicast IP address, the receiver **MUST** notify the underlying network of its interest to receive multicast packets to this address. In an IP multicast network, this is done by sending an IGMP/MLD join on (S,G) in the case of single-source multicast. If the IP destination address in the FC_ANNOUNCE frame is the unicast address of the receiver, it means that the source uses unicast-duplication. The receiver **MAY** also wait to receive the FC_KEY frame of the corresponding multicast flow to avoid receiving packets that it will not be able to process before receiving the required TLS keys.

6. Multicast flow membership management

Thanks to the FC_STATE frame, a Flexicast QUIC source knows the set of receivers listening to a specific multicast flow. Receivers **MAY** decide, at any point in time during the communication, to leave the multicast flow.

6.1. Receiver-side management

Receivers leave a multicast flow by sending an FC_STATE frame with the LEAVE action. Upon reception of this frame, the Flexicast QUIC source **MUST** remove the receiver from the members of the multicast flow, and it **MAY** continue transmitting data through the unicast path with the receiver; the source **MAY** also decide to close the connection with the receiver, depending on the

application. The receiver **MUST** drop any path state for the multicast flow regarding this multicast flow. The finite-state machine on the receiver-side **MUST** be reset to the same state as when it received the FC_ANNOUNCE for the first time.

A receiver that previously left a multicast flow **MAY** attempt to join again the same flow, or another flow by restarting the phases from Section 5.2.

6.2. Source-side management

At any point in time, the Flexicast QUIC source **MAY** decide to unilaterally remove a receiver from a multicast flow, by sending an FC_STATE frame with the LEAVE action.

There are two reasons to decide on the source-side to remove receivers from the multicast flow: - A receiver is a bottleneck on the multicast flow, i.e., the bit-rate of the multicast flow becomes too low because of this receiver. In that case, the bottleneck receiver is removed from the specific multicast flow. The source can continue distributing the content either through the unicast path with this receiver, or by letting the receiver join another multicast flow. - A receiver experiences too many losses. Receivers send feedback to the source. Too many losses degrade the quality of experience for the application. To recover from such losses, there is a need for retransmissions, which can take up to several RTTs. The source **SHOULD** decide to remove the receiver from the multicast flow and continue receiving data through unicast, even temporarilly. A reason why it would be better to continue through the unicast path is that the underlying IP multicast tree may be failing.

7. Reliability

The reliability mechanism of Flexicast QUIC uses the standard QUIC mechanisms. This section only details the reliability mechanisms for frames sent on the multicast flow. This document does not modify the reliability mechanism defined in [RFC9002] for packets sent on the unicast path.

Receivers regularly send PATH_ACK frames back to the source on their unicast path. Receivers **MUST** send PATH_ACK frames for packets received on the multicast flow on their unicast path with the source. A Flexicast QUIC source receiving an PATH_ACK from a receiver on the multicast flow **MUST** close the connection with this receiver with an error of type FC_PROTOCOL_VIOLATION. The PATH_ACK frames sent by the receivers on their unicast path include the `path_id` field to refer to the multicast flow.

A major change between [RFC9002], and to some extent [MULTIPATH-QUIC] is that a Flexicast QUIC source receives acknowledgments from multiple receivers for the same data. A Flexicast QUIC source must wait for the acknowledgment from all receivers listening to the multicast flow before releasing state for the transmitted data. An acknowledgment-aggregation mechanism **MUST** be implemented, at least on the Flexicast QUIC source, to advance its state when all members of the multicast flow sent PATH_ACK frames to acknowledge data sent on the multicast flow.

Bottleneck or malicious receivers may slow down, and even block the transmission of data, thus impacting the quality of service for other receivers. Similarly to [RFC9002], a Flexicast QUIC source may decide to stop the communication with a specific receiver if it does not regularly receive PATH_ACK frames from this receiver. A Flexicast QUIC source **SHOULD** remove bottleneck-inducing or malicious receivers from the multicast flow in that case, and **MAY** continue transmitting data through the unicast path with these receivers instead, thus relying on [RFC9002].

The Flexicast QUIC source is responsible to decide whether to send retransmitted frames on the multicast flow to all receivers, or specifically to receivers individually on their unicast path. Receivers are not impacted by this choice since the [MULTIPATH-QUIC] specification allows to retransmit frames on another path than the initial path on which they were sent. The first case would be more efficient if multiple receivers lost the same packet; the second case is more interesting for isolated losses to avoid healthy receivers receiving duplicate frames.

8. Congestion Control

There are currently several possibilities regarding congestion control for Flexicast QUIC. A first idea is to maintain constant bit-rate delivery and exposing multiple multicast flows operating at different bit-rates. As such, if a receiver sees degradation in its communication (e.g., congestion or increased delay in the network), it may switch to a lower bit-rate multicast flow. However, this idea does not solve the problem of increasing the congestion in the network.

The other idea outlines the possibility to take into account all receiver-specific bit-rate to chose the overall bit-rate on the multicast flow, e.g., by requiring that the multicast flow bit-rate is the lowest bit-rate among all receivers listening to this multicast flow. Since receivers regularly send PATH_ACK frames, it is possible for the source to adjust a per-receiver bit-rate (or congestion window) and use the minimum value as the value for the multicast flow. Of course, such method paves the way to malicious receivers decreasing on-purpose the multicast flow bit-rate. Applications **SHOULD** provide to a Flexicast QUIC source a "minimum bit-rate" to ensure a minimum quality of service for the receivers. Malicious or bottleneck receivers with a per-receiver bit-rate below this minimum bit-rate **SHOULD** be removed from the multicast flow membership. Since the Flexicast QUIC source can unilaterally evict such receivers, it can adjust the multicast flow bit-rate without considering these receivers.

A mix of both approaches is also possible. A Flexicast QUIC source may expose multiple multicast flow, each operating at different bit-rate windows. For example, a first window would operate at a minimum bit-rate of 2 Mbps and a maximum bit-rate of 5 Mbps; another multicast flow would operate between 5 Mbps and 10 Mbps,... Receivers falling below the 2 Mbps bit-rate **SHOULD** be evicted from flexicast delivery; a Flexicast QUIC source **MAY** decide to continue the delivery through the unicast path.

9. Flow Control

Similarly to the congestion control, an aggregated flow control mechanism extends the standard QUIC flow control mechanism from [QUIC-TRANSPORT]. The multicast flow **MUST** respect the flow control limits of active members. Receivers actively listening to the multicast flow **MUST** send MAX_DATA and MAX_STREAM_DATA on their unicast path, similarly to [QUIC-TRANSPORT]. The multicast flow aggregates all flow control limits from active members and **MUST** set its limits to the minimum among all limits from the receivers.

To ensure that the flow control does not violates any limit, the flow control **MAY** use values from different active receivers to update its flow control limits. For example with three receivers `R0`, `R1`, and `R2`, the multicast flow may update its max data using the value advertised by `R0`, its max stream data for stream id X using the value advertised by `R1`, and its max stream data for stream id Y using the value advertised by `R2`.

The source **MAY** remove bottleneck receivers from the set of active members if such receivers fail to update their flow control to ensure a minimum quality of service. In that case, the source **MAY** continue transmiting data to these receivers on their unicast path, again as a fall-back on unicast QUIC as defined in [QUIC-TRANSPORT].

Because the flow control limits are shared between multiple paths within the same connection, a receiver which previously fall backed on unicast delivery **MAY** rejoin a multicast flow if its flow control limits have been updated.

10. Packet protection

Packet protection for QUIC version 1 is specified in Section 5 of [QUIC-TLS]. Section 4 of [MULTIPATH-QUIC] further expands this design when multiple paths with different packet number spaces are used. Because the multicast flow is shared among multiple receivers, this document modifies the design from Section 4 of [MULTIPATH-QUIC].

10.1. Protection Keys

This document extends the design from [MULTIPATH-QUIC] by requiring that all multicast flows use different TLS protection keys. Of course, these protections keys **MUST** be different from the protection keys derived during the handshake between the source and any receiver.

For each multicast flow, the Flexicast QUIC source derives new random TLS keys that will be used to encrypt and decrypt the packets. The Flexicast QUIC source **MUST** derive these keys randomly, simulating a QUIC connection establishment alone. The master secret and used algorithm is sent through the FC_KEY frame (Section 11.3) on the unicast path to receivers joining a multicast flow. The master secret and algorithm are used to derive the TLS decryption keys on the receivers.

Since no explicit path probing phase is used in Flexicast QUIC, the receiver can use the dedicated protection key context whenever receiving a packet from the multicast flow directly after receiving the FC_KEY frame from the source.

10.2. Nonce Calculation

Section 4 of [MULTIPATH-QUIC] expands the computation of the Nonce from Section 5.3 of [QUIC-TLS] to integrate the least significant 32 bits of the Path ID to guarantee its uniqueness. A multicast flow is shared among multiple receivers simultaneously, thus requiring that all receivers share the same Path ID for the same multicast flow. Since receivers are allowed to dynamically change the multicast flows they listen, it is impossible to ensure that all receivers use the same Path ID for the same multicast flow.

However, since each multicast flow uses its own set of TLS keys, the computation of the Nonce is decorelated between any pair of multicast flows, and between any multicast flow and any unicast path with a receiver. It is therefore not mandatory anymore to use the Path ID to ensure the uniqueness of the Nonce. As such, this document removes the Path ID from the computation of the Nounce when sending packets on the multicast flows.

10.3. Key Update

TODO in a future version of the draft.

11. New Frames

All frames defined in this document **MUST** only be sent in 1-RTT packets.

If an endpoint receives a flexicast-specific frame in a different packet type, it **MUST** close the connection with an error of type FRAME_ENCODING_ERROR.

Receipt of flexicast-specific frames related to a Flexicast Flow ID that is not unknown by endpoint **MUST** be treated as a connection error of type FC_PROTOCOL_VIOLATION.

If an endpoint receives a flexicast-specific frame with a Flexicast Flow ID that it cannot process anymore (e.g., the multicast flow might have been abandoned), it **MUST** silently ignore the frame.

The new frames introduced below are for control-specific purpose only, and **MUST NOT** be sent on the Multicast flow. Receipt of any of the following frames on the Multicast flow **MUST** be trated as a connection error of type FC_PROTOCOL_VIOLATION.

11.1. FC_ANNOUNCE frame

The FC_ANNOUNCE frame informs the receiver that a Multicast flow is available or has been updated. FC_ANNOUNCE frames **MUST NOT** be sent by the receiver. A Flexicast QUIC source receiving an FC_ANNOUNCE frame **MUST** close the connection with a connection error of type FC_PROTOCOL_VIOLATION.

FC_ANNOUNCE frames are formatted as shown in Figure 4.

```
FC_ANNOUNCE Frame {
    Type (i) = TBD-00,
    Length (8),
    Flexicast Flow ID (8..160),
    Sequence number (i),
    IP Version (8),
    Source IP (32, 128),
    Group IP (32, 128),
    UDP Port (16),
    Ack delay timer (64),
}
```

Figure 4: FC_ANNOUNCE Frame Format

FC_ANNOUNCE frames contain the following fields:

Length: An 8-bit unsigned integer containing the length of the Flexicast Flow ID. Values less than 1 and greater than 20 are invalid and **MUST** be treated as a connection error of type FRAME_ENCODING_ERROR.

Flexicast Flow ID: A Flexicast Flow ID of the specified length.

Sequence number: A variable-length integer indicating the sequence of the frame. The number is monotonically increasing within a QUIC connection and is chosen by the sender. It helps the receiver to order FC_ANNOUNCE frames by recency. A receiver **SHOULD** ignore frames with a Sequence Number lower or equal to the highest Sequence Number received.

IP Version: An 8-bit unsigned integer containing the version of IP used to advertise the Source IP and Group IP. Values different than 4 (for IPv4) and 6 (IPv6) are invalid and **MUST** be treated as a connection error of type FRAME_ENCODING_ERROR.

Source IP: The IP address of the multicast source, used for Single-Source Multicast.

Group IP: Either an IP multicast address or the address of the receiver.

UDP Port: The UDP destination port.

Ack delay timer: A 64-bit unsigned integer containing the delay, in ms, between two acknowledgments from a receiver.

FC_ANNOUNCE frames are ack-eliciting. If a packet containing an FC_ANNOUNCE frame is considered lost, the peer **SHOULD** repeat it.

Sources are allowed to send multiple times FC_ANNOUNCE frames with an increasing sequence number for the same Flexicast Flow ID. New FC_ANNOUNCE frames **MAY** contain updated information, e.g., a new Ack delay timer.

Sources are allowed to advertise multiple Multicast flows by sending multiple parallel FC_ANNOUNCE frames with distinct Flexicast Flow IDs. The Sequence number is linked to a specific Flexicast Flow ID. The same Sequence number can be used for two distinct Flexicast Flow IDs.

A Flexicast QUIC can withdraw a multicast flow by sending an FC_ANNOUNCE frame with null Source IP and Group IPs and identifying the multicast flow using the Flexicast Flow ID. Upon reception of a new FC_ANNOUNCE frame updating information of an existing multicast flow with an increased sequence number compared to the last received FC_ANNOUNCE frame, a receiver **MUST** update multicast flow information.

11.2. FC_STATE frame

The FC_STATE frame informs the endpoint of the state of the Flexicast receiver in the Multicast flow. FC_STATE frames **MAY** be sent by both endpoints (i.e., receiver and source).

FC_STATE frames are formatted as shown in Figure 5.

```
FC_STATE frame {
    Type (i) = TDB-01,
    Length (8),
    Flexicast Flow ID (8..160),
    Sequence number (i),
    Action (u64),
}
```

Figure 5: FC_STATE Frame Format

FC_STATE frames contain the following fields:

Length: An 8-bit unsigned integer containing the length of the Flexicast Flow ID. Values less than 1 and greater than 20 are invalid and **MUST** be treated as a connection error of type FRAME_ENCODING_ERROR.

Flexicast Flow ID: The Flexicast Flow ID of the Multicast flow that this frame relates to.

Sequence number: The monotically increasing sequence number related to the advertised Flexicast Flow ID.

Action: The bit-encoded action, defined in Section Section 11.2.1.

FC_STATE frames are ack-eliciting. If a packet containing an FC_STATE frame is considered lost, the peer **SHOULD** repeat it.

For a given Multicast flow (i.e., identical Flexicast Flow ID), both endpoints use their own Sequence number.

A receiver sending an FC_STATE frame informs the source of its status regarding the Multicast flow indicated by the Flexicast Flow ID. The source **MAY** also send FC_STATE frames to a receiver to unilaterally change the status of the receiver within the Multicast flow indicated by the Flexicast Flow ID.

11.2.1. FC_STATE actions

This section lists the defined Actions encoded in an FC_STATE frame. An endpoint receiving an unknown value **MUST** treat it as a connection error of type FC_PROTOCOL_VIOLATION.

JOIN (0x01): The receiver joins the Multicast flow.

LEAVE (0x02): The receiver leaves the Multicast flow.

READY (0x03): The receiver is ready to receive content on the Multicast flow.

The JOIN and READY actions are receiver-specific. These actions **MUST NOT** be sent inside an FC_STATE frame sent by the source. A receiver receiving an FC_STATE frame with any of the following actions **MUST** treat it as a connection error of type FC_PROTOCOL_VIOLATION. The action LEAVE **MAY** be sent by both the receiver and the source, as detailed in Section 6.

11.3. FC_KEY frame

The FC_KEY frame informs a receiver of the security keys used on a Multicast flow joined by the receiver. FC_KEY frames **MUST NOT** be sent by the receiver. A Flexicast QUIC source receiving an FC_KEY frame **MUST** close the connection with a connection error of type FC_PROTOCOL_VIOLATION.

FC_KEY frames are formatted as shown in Figure 6.

```
FC_KEY Frame {
    Type (i) = TDB-02,
    Length(8),
    Flexicast Flow ID(8..160),
    Sequence number (i),
    Packet number (i),
    Key length (i),
    Key (..),
    Algorithm (64),
}
```

Figure 6: FC_KEY Frame Format

FC_KEY frames contain the following fields:

Length: An 8-bit unsigned integer containing the length of the Flexicast Flow ID. Values less than 1 and greater than 20 are invalid and **MUST** be treated as a connection error of type FRAME_ENCODING_ERROR.

Flexicast Flow ID: The Flexicast Flow ID of the Multicast flow that this frame relates to.

Sequence number: The monotically increasing sequence number related to the advertised Flexicast Flow ID.

Packet number: The first packet number that the receiver must receive with this key,

Key length: A var-int indicating the length of the security key.

Key: Byte-sequence of the decryption key of the Multicast flow.

Algorithm: The bit-encoded algorithm used for decryption, defined in Section Section 11.3.1.

FC_KEY frames are ack-eliciting. If a packet containing an FC_STATE frame is considered lost, the peer **SHOULD** repeat it.

The source **MAY** send new FC_KEY frames with an increased sequence number to notify a new decryption key. This mechanism can be used to provide backward and forward secrecy with dynamic Flexicast groups.

11.3.1. FC_KEY algorithms

The algorithms and their encoding follow [RFC8446] and {QUIC-TLS}. TODO: expand this section.

12. Discussion

This document has defined a simple extension to Multipath QUIC that enables a QUIC connection to simulatenously use unicast paths and multicast trees to deliver the same data to a set of receivers. The proposed protocol can be extended in different ways and improvements will be proposed in separate documents. We briefly describe some of these possible extensions.

This version of Flexicast QUIC uses the existing QUIC mechanisms to retransmit lost frames. It is well known that Forward Erasure Correction can improve the performance of multicast transmission when losses occur. Several authors have proposed techniques to add Forward Erasure Correction to QUIC [QUIRL], [rQUIC], [I-D.draft-michel-quic-fec]. FEC can be sent a priori to enable receivers to recover from different packet losses without having to wait for retransmissions or a posteriori by sending a repair symbol that enables different receivers to recover different lost frames.

This version of Flexicast QUIC uses a key shared between the source and all receivers to authenticate and encrypt the data sent by the source over the multicast tree. A malicious receiver who has received the shared key could inject fake data over the multicast tree provided that it can spoof the IP address of the source. Techniques have been proposed to authenticate the frames sent by the source [I-D.draft-krose-multicast-security]. Subsequent documents will detail how Flexicast QUIC can be extended to support such techniques.

Multipath QUIC assumes that the congestion control mechanism operates per path. For Flexicast QUIC, a different congestion control mechanism will be required for the unicast paths and the multicast tree. For the former, the QUIC congestion control mechanisms [RFC9002] are applicable. For the latter, multicast specific congestion control mechanisms such as [RFC4654] will be required. The current prototype uses a variant of the CUBIC congestion control.

13. Security Considerations

This section highlights security considerations when operating a Flexicast QUIC communication between a single source and multiple receivers. Since a unique multicast flow is shared among multiple receivers, additional threats must be addresses compared to a one-to-one (Multipath) QUIC connection. The security considerations when falling-back on unicast are similar to Section 10 of [MULTIPATH-QUIC].

TODO: this section will be expanded in future versions of this document.

13.1. Sending the TLS key secrets in FC_KEY frames

TODO: discuss here the potential issues by sending the TLS key secrets in the FC_KEY frames. I don't think there is any problem since the secrets are sent through a secure channel.

13.2. Malicious Receivers in a Flexicast Flow

Malicious receivers may listen to a multicast flow in the presence of healthy receivers. This has several impacts that are addressed in this section.

13.2.1. Cycling Between Joins and Leaves

A malicious receiver may issuing cycles of FC_STATE with JOINs and LEAVEs actions to the source, thus updating the state of the Flexicast QUIC source. Since a Flexicast QUIC source can decide to unilaterally remove receivers from a multicast flow, the source can decide to reject FC_STATE JOINs from a malicious receiver. The source **MAY** send an FC_STATE frame withdrawing a multicast flow specifically for this receiver to avoid the receiver from perpetually sending FC_STATE JOINs frames. Additionally, applications **SHOULD** provide a mechanism to avoid such receivers to periodically join and leave the same multicast flow to saturate the Flexicast QUIC source. The connection with such malicious receiver will fall-back on unicast delivery, thus relying on [QUIC-TRANSPORT] to deal with it.

13.2.2. Intentionally Decreasing the Flexicast Flow Performance

Section 6.2 mentions two reasons that could include malicious receivers wishing to degrade the overall perfomance of communication through the multicast flow. By letting the source unilaterally decide to remove receivers from a multicast flow, the impact of malicious receivers is limited.

14. IANA Considerations

IANA is requested to assign three new QUIC frame types from the "QUIC Frame Types" registry available at https://www.iana.org/assignments/quic/quic.xhtml#quic-frame-types

- TBD-00 for the FC_ANNOUNCE frame defined in Section 11.1
- TBD-01 for the FC_STATE frame defined in Section 11.2
- TBD-02 for the FC_KEY frame defined in Section 11.3

IANA is requested to assign a new QUIC transport parameter from the "QUIC Transport Parameters" registry available at https://www.iana.org/assignments/quic/quic.xhtml#quic-transport

- TBD-03 for the flexicast_support transport parameter defined in Section 4

15. References

15.1. Normative References

[MULTIPATH-QUIC] Liu, Y., Ma, Y., De Coninck, Q., Bonaventure, O., Huitema, C., and M. Kühlewind, "Managing multiple paths for a QUIC connection", Work in Progress, Internet-Draft, draft-ietf-quic-multipath-20, 20 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-20>.

[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>.

[QUIC-TRANSPORT] 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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

15.2. Informative References

[DIOT]

Diot, C., Levine, B., Lyles, B., Kassem, H., and D. Balensiefen, "Deployment issues for the IP multicast service and architecture", Institute of Electrical and Electronics Engineers (IEEE), IEEE Network vol. 14, no. 1, pp. 78-88, DOI 10.1109/65.819174, 2000, <https://doi.org/10.1109/65.819174>.

[FCQUIC] Navarre, L., "AAA", ACM SIGCOMM Computer Communication Review (CCR), March 2025, <https://dial.uclouvain.be/pr/boreal/object/boreal:301111>.

[I-D.draft-krose-multicast-security] Rose, K., Franke, M., and J. Holland, "Security and Privacy Considerations for Multicast Transports", Work in Progress, Internet-Draft, draft-krose-multicast-security-07, 7 May 2025, <https://datatracker.ietf.org/doc/html/draft-krose-multicast-security-07>.

[I-D.draft-michel-quic-fec] Michel, F. and O. Bonaventure, "Forward Erasure Correction for QUIC loss recovery", Work in Progress, Internet-Draft, draft-michel-quic-fec-01, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-michel-quic-fec-01>.

[I-D.jholland-quic-multicast-08] Holland, J., Pardue, L., Franke, M., and K. Rose, "Multicast Extension for QUIC", Work in Progress, Internet-Draft, draft-jholland-quic-multicast-08, 2 January 2026, <https://datatracker.ietf.org/doc/html/draft-jholland-quic-multicast-08>.

[I-D.pardue-quic-http-mcast] Pardue, L., Bradbury, R., and S. Hurst, "Hypertext Transfer Protocol (HTTP) over multicast QUIC", Work in Progress, Internet-Draft, draft-pardue-quic-http-mcast-11, 4 July 2022, <https://datatracker.ietf.org/doc/html/draft-pardue-quic-http-mcast-11>.

[QUIRL] Michel, F. and O. Bonaventure, "QUIRL: Flexible QUIC Loss Recovery for Low Latency Applications", Institute of Electrical and Electronics Engineers (IEEE), IEEE/ACM Transactions on Networking vol. 32, no. 6, pp. 5204-5215, DOI 10.1109/tnet.2024.3453759, December 2024, <https://doi.org/10.1109/tnet.2024.3453759>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/rfc/rfc1112>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <https://www.rfc-editor.org/rfc/rfc4607>.

[RFC4654] Widmer, J. and M. Handley, "TCP-Friendly Multicast Congestion Control (TFMCC): Protocol Specification", RFC 4654, DOI 10.17487/RFC4654, August 2006, <https://www.rfc-editor.org/rfc/rfc4654>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/rfc/rfc9293>.

[rQUIC] Garrido, P., Sanchez, I., Ferlin, S., Aguero, R., and O. Alay, "rQUIC: Integrating FEC with QUIC for Robust Wireless Communications", IEEE, 2019 IEEE Global Communications Conference (GLOBECOM), DOI 10.1109/globecom38437.2019.9013401, December 2019, <https://doi.org/10.1109/globecom38437.2019.9013401>.

Appendix A. Existing implementation and initial results.

The design of Flexicast QUIC is presented in a scientific paper, accepted at ACM SIGCOMM Computer Communication Review (CCR) [FCQUIC]. We provide a proof of concept implementation of Flexicast QUIC, relying on Cloudflare quiche and the multipath extension. The source code of Flexicast QUIC is freely available for experimentations at https://github.com/IPNetworkingLab/flexicast-quic.

Acknowledgments

Louis Navarre was partially funded as an F.R.S-FNRS Research Fellow. This work has been partially supported by the Walloon Region as part of the funding of the FRFS-WEL-T strategic axis.

We thank Maxime Piraux and Gorry Fairhurst for their valuable reviews.

Authors' Addresses

Louis Navarre
UCLouvain
Belgium
Email: louis.navarre@uclouvain.be

Olivier Bonaventure
UCLouvain & WELRI
Belgium
Email: olivier.bonaventure@uclouvain.be