DEMO: Towards an Internet Deployment of Flexible Multicast QUIC

Louis Navarre
UCLouvain, Belgium
louis.navarre@uclouvain.be

Olivier Bonaventure
UCLouvain & WEL-RI, Belgium
olivier.bonaventure@uclouvain.be

ABSTRACT

Despite their scalability benefits, multicast protocols are not widely deployed and are confined to intra-domain use cases such as IPTV. Several factors contribute to this reluctance to deploy multicast on the Internet. Among them, the *chicken-and-egg* and *all-or-nothing* problems hindered the emergence of inter-domain multicast. Recent advancements within the IETF suggest solutions to these issues, such as Automatic Multicast Tunneling (AMT) and flexible multicast extensions to the QUIC transport protocol (FCQUIC). This *demo* leverages AMT and Flexicast QUIC to enable researchers to conduct large-scale inter-domain multicast measurement campaigns. We provide Docker images of FCQUIC receivers that can be connected to our FCQUIC source, which any Internet host can reach using AMT. We will also publish our setup code to motivate researchers to instantiate their own FCQUIC source application with AMT.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Transport protocols**.

KEYWORDS

Multicast, QUIC, Flexicast

ACM Reference Format:
Louis Navarre and Olivier Bonaventure. 2025. DEMO: Towards an Internet Deployment of Flexible Multicast QUIC. In *ACM SIGCOMM 2025 Conference (SIGCOMM Posters and Demos '25), September 8–11, 2025, Coimbra, Portugal*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3744969.3748403

1 INTRODUCTION

Deering initially proposed IP Multicast in the late 80s [4, 5]. It enables a source to efficiently send the same data to multiple receivers, relying on (multicast-capable) routers to replicate packets to reach all receivers, forming a distribution tree. Despite its clear scalability benefits, operators did not deploy multicast on the Internet due to several issues [6], including the *chicken-and-egg* problem, which prevents the emergence of use cases until an inter-domain multicast network is set up, and conversely. Similarly, application developers prefer unicast because of the *all-or-nothing* nature of existing multicast transport protocols, which requires applications to develop their own unicast *fall-back* mechanism in case a receiver cannot join the multicast tree. Nowadays, multicast is confined to intra-domain use cases such as IPTV [16, 17], enterprise software update [18], and financial applications [3].

Despite these limitations, multicast remains attractive due to its potential for scalability. Indeed, Akamai, one of the world's largest Content Delivery Network (CDN) providers, reported that between 20 % to 50 % of its traffic (mostly VOD/Live streaming) could benefit from multicast forwarding to alleviate the pressure in its network [12]. In 2022, Akamai reached peak traffic of 250 Tbps [1], showing the substantial potential benefits of multicast. These reasons pushed the IETF to specify on modern solutions to the aforementioned issues of multicast, such as Automatic Multicast Tunneling (AMT) [2]. AMT permits incremental multicast deployment, as non-multicast receivers can still benefit from multicast content through a unicast tunnel. We leverage an open-source implementation of AMT [10] in software.

In parallel, researchers explored ways to extend the QUIC transport protocol [14] with multicast extensions [13, 19, 21]. These approaches suggest exposing a *shared* QUIC communication channel between the source and the receivers and a *per-receiver* connection for control and *fall-back* data distribution. We focus on the second design, Flexicast QUIC (FCQUIC) [19], and leverage our open-source implementation of this extension[1] [20]. FCQUIC solves the *all-or-nothing* problem of multicast as the application leverages efficient multicast when possible, and *seamlessly* falls back on unicast otherwise within the same transport protocol.

With Automatic Multicast Tunneling [2] and Flexicast QUIC [20], it becomes possible to reconsider deploying multicast use cases on the Internet. In this *demo*, we leverage AMT and FCQUIC to initiate new multicast measurement campaigns over the Internet. These large-scale measurements will help to identify the scalability benefits and limits of both FCQUIC and AMT, as well as motivate the networking community to reconsider multicast communication for one-to-many use cases. As a primary step towards this direction, we build a *push-based* RSS feeder application of the Linux kernel mailing list and expose a public FCQUIC server distributing RSS feeds to FCQUIC receivers. We provide Docker images of FCQUIC receivers to connect to our public server and receive multicast packets using AMT. While this application is simple, we aim to explore the limits of an FCQUIC server to scale to larger and heterogeneous audiences on the real Internet. The *docker* images are public, and all information is available at: https://forge.uclouvain.be/louisna/fcquic-deployment-sitcom-2025-demo.git.

2 BACKGROUND

Automatic Multicast Tunneling is a tunneling protocol aiming forward multicast packets from a multicast-capable network to

[1]The implementation is not a contribution of this *demo*.

SIGCOMM Posters and Demos '25, September 8–11, 2025, Coimbra, Portugal

ACM ISBN 979-8-4007-2026-0/2025/09…$15.00
https://doi.org/10.1145/3744969.3748403

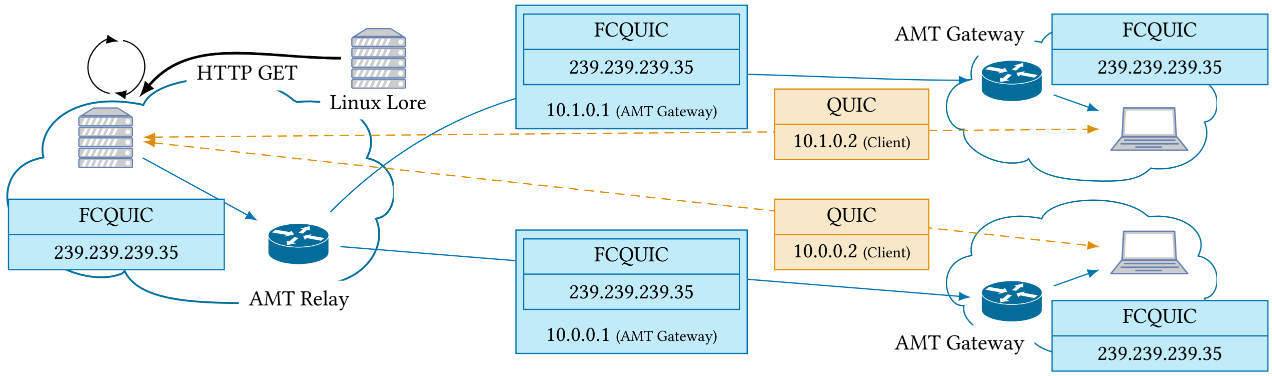

Figure 1: Deployment architecture of Flexicast QUIC with Automatic Multicast Tunneling on the Internet.

receivers lacking such multicast connectivity. As illustrated in Figure 1, AMT combines a *Relay* with one or several *Gateways*. The Relay lies in the source, multicast-capable network and listens to Gateway requests to join specific multicast groups. The Gateway stands in the receiver's network, potentially on the same host. It establishes a tunnel with the Relay and forwards IGMP/MLD requests from the receiver to the Relay, which in turn forwards the requests to the source of its multicast-capable network (e.g., using PIM), thus adding the Relay to the multicast distribution tree in this network. The Relay encapsulates the multicast packets in UDP to reach the Gateway; the latter removes the encapsulation and forwards the multicast packet to the receiver.

Flexicast QUIC is an extension of QUIC [14] and Multipath QUIC [15] providing *flexible multicast* to application. It combines *unidirectional* flexicast flows *shared* with multiple receivers and one *per-receiver bidirectional* unicast path as a secure control and fallback mechanism. By leveraging Multipath QUIC [15], an FCQUIC receiver sends acknowledgment on its unicast path to the source; the latter can decide to retransmit lost frames either on the flexicast flow (e.g., if multiple receivers lost the corresponding packet) or on the unicast path to reach specific subsets of receivers. Packets sent on the shared flexicast flow can be sent using a multicast IP address. Receivers lying in an ill-configured multicast network (e.g., due to a bad configuration) can fall back on their unicast path without impacting the remaining group members.

3 INTERNET DEPLOYMENT

Our source is a small *push-based* RSS feeder that distributes the messages received on the Linux kernel Lore mailing list. The Kernel Lore already provides mail news through an RSS feed, but host applications are responsible for pulling the feed. As illustrated on Figure 1, our Flexicast QUIC source pulls the RSS feed every five seconds, using the *If-Modified-Since* HTTP header [8] to avoid redundant transfer. Upon new feed, the FCQUIC source sends the content in a new QUIC stream on the *flexicast flow*, i.e., the shared unidirectional channel towards the receivers. New receivers first establish a QUIC connection with the source and upgrade it to Flexicast [19]. The receiver's AMT Gateway establishes the tunnel with the Relay and receives hereafter encapsulated multicast packets. Our Docker image embeds the IP address of our AMT Relay for simplicity, but this process can be automated using DRIAD [11].

Initial measurement results. We host the server in our campus network and performed initial measurements with 9 clients to show

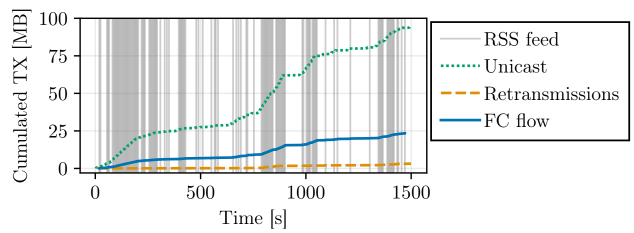

Figure 2: Cumulated sent bytes on the server, comparing unicast QUIC with Flexicast QUIC (`FC flow`). `Retransmissions` contains the unicast-retransmitted data when losses occur on the flexicast flow.

the benefits of multicast from the server's viewpoint: 5 clients listen to the flexicast flow (i.e., receive multicast packets through AMT), and 4 others receive the content through unicast: 3 receivers (2 multicast, 1 unicast) are hosted on Cloudlab (USA) [7]; 1 multicast receiver in South Africa; the remaining receivers are located on the same campus as the source. Figure 2 highlights the benefits of multicast for the server regarding the bytes sent; the source sends $\sim 5\times$ fewer bytes using Flexicast QUIC. We occasionally notice retransmissions of packets lost on the flexicast flow. This initial result confirms the successful deployment of our RSS feed application.

4 CONCLUSION AND FUTURE WORK

This *demo* aims to initiate measurements of the new Flexicast QUIC (FCQUIC) on the Internet. We provide Docker images of an FCQUIC receiver that connects to our source and receives multicast data using Automatic Multicast Tunneling (AMT). We will also release our source code, enabling researchers to instantiate their own FCQUIC source and AMT Relay. This simple application serves as the starting point for larger-scale measurement campaigns of multicast-enabled use cases, and we hope to engage the scientific community in these campaigns. Our implementation of Flexicast QUIC is already freely available [20]. This paper's contributions relate to the RSS feeder application and deployment of an FCQUIC source. In future work, we will evaluate FCQUIC's deployment with video stream applications, integrating into TreeDN [9], a multicast tree-based delivery network for live streaming applications. We will leverage the community's interest to perform large-scale measurement campaigns, and hope to motivate researchers to deploy their own Flexicast QUIC/AMT instances towards the widespread adoption of multicast on the Internet.

REFERENCES

[1] Akamai. 2022. Oops, we did it again. (2022). https://www.linkedin.com/pulse/oops-we-did-again-akamai-technologies.

[2] Gregory Bumgardner. 2015. Automatic Multicast Tunneling. RFC 7450. https://doi.org/10.17487/RFC7450

[3] Cisco. 2008. Trading Floor Architecture. (2008). https://www.cisco.com/c/en/us/td/dEstringocs/solutions/Verticals/Trading_Floor_Architecture-E.html

[4] Stephen E Deering. 1988. Multicast routing in internetworks and extended LANs. In *Symposium proceedings on Communications architectures and protocols*. 55–64.

[5] Stephen E Deering. 1989. Host extensions for IP multicasting. RFC 1112. https://doi.org/10.17487/RFC1112

[6] Christophe Diot, Brian Neil Levine, Bryan Lyles, Hassan Kassem, and Doug Balensiefen. 2000. Deployment issues for the IP multicast service and architecture. *IEEE network* 14, 1 (2000), 78–88.

[7] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. 1–14. https://www.flux.utah.edu/paper/duplyakin-atc19

[8] Roy T. Fielding and Julian Reschke. 2014. Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. RFC 7232. https://doi.org/10.17487/RFC7232

[9] Lenny Giuliano, Chris Lenart, and Rich Adam. 2025. TreeDN: Tree-Based Content Delivery Network (CDN) for Live Streaming to Mass Audiences. RFC 9706. https://doi.org/10.17487/RFC9706

[10] Jake Holland. [n. d.]. Fork of the AMT protocol reference implementation (RFC 7450) from http://www.utdallas.edu/ ksarac/amt/. ([n. d.]). https://github.com/GrumpyOldTroll/amt.git Consulted: May 14th, 2025.

[11] Jake Holland. 2020. DNS Reverse IP Automatic Multicast Tunneling (AMT) Discovery. RFC 8777. https://doi.org/10.17487/RFC8777

[12] Jake Holland. 2020. IP Multicast: Next steps to make it real. (2020). NANOG79, available from https://youtu.be/2aihLUb1elg?list=PLO8DR5ZGla8i-aVXtTFRZ6l7BRBvYdrkO.

[13] Jake Holland, Lucas Pardue, and Max Franke. 2025. *Multicast Extension for QUIC*. Internet-Draft draft-jholland-quic-multicast-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-jholland-quic-multicast/06/ Work in Progress.

[14] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. https://doi.org/10.17487/RFC9000

[15] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, and Mirja Kühlewind. 2025. *Multipath Extension for QUIC*. Internet-Draft draft-ietf-quic-multipath-14. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/14/ Work in Progress.

[16] Julien Maisonneuve, Muriel Deschanel, Juergen Heiles, Wei Li, Hong Liu, Randy Sharpe, and Yiyan Wu. 2009. An overview of IPTV standards development. *IEEE Transactions on broadcasting* 55, 2 (2009), 315–328.

[17] Marco Mellia and Michela Meo. 2010. Measurement of IPTV traffic from an operative network. *European Transactions on Telecommunications* 21, 4 (2010), 324–336.

[18] Microsoft. 2022. Use multicast to deploy Windows over the network with Configuration Manager. (2022). https://learn.microsoft.com/en-us/mem/configmgr/osd/deploy-use/use-multicast-to-deploy-windows-over-the-network

[19] Louis Navarre and Olivier Bonaventure. 2024. *Flexicast QUIC: combining unicast and multicast in a single QUIC connection*. Internet-Draft draft-navarre-quic-flexicast-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-navarre-quic-flexicast/00/ Work in Progress.

[20] Louis Navarre, Quentin De Coninck, Tom Barbette, and Olivier Bonaventure. 2025. Taking the Best of Multicast and Unicast with Flexicast QUIC. *ACM SIGCOMM Computer Communication Review* 55, 2 (2025). Postprint.

[21] Lucas Pardue, Richard Bradbury, and Sam Hurst. 2022. *Hypertext Transfer Protocol (HTTP) over multicast QUIC*. Internet-Draft draft-pardue-quic-http-mcast-11. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-pardue-quic-http-mcast/11/ Work in Progress.