Optimization of QUIC Congestion Control with ECN

Rico Finkbeiner, Marcel Kempf*, Benedikt Jaeger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: rico.finkbeiner@tum.de, kempfm@net.in.tum.de, jaeger@net.in.tum.de

***Abstract*—QUIC plays an important role in today's Internet by providing several benefits over TCP. In this paper, we explain how ECN can be used to optimize QUIC's congestion control. ECN tries to avoid retransmissions by explicitly notifying the sender about congestion in the network instead of silently dropping packets. When using QUIC, this can be done by including ECN counts in the ACK frames to mirror information about incipient congestions back to the sender, who can then reduce its sending rate. We also take a look at the support of ECN with QUIC in the Internet based on related work. ECN with QUIC is currently barely used, mainly because of missing support in common QUIC implementations and failures in the ECN validation stage.**

***Index Terms*—QUIC, ECN, congestion control**

1. Introduction

QUIC, a protocol introduced by Google [1], provides many benefits over the Transmission Control Protocol (TCP). By providing a zero round-trip time (0-RTT) handshake and avoiding head-of-line blocking delays, QUIC significantly reduces latency [1], [2]. Among other reasons, this has led to a wide adoption of QUIC in the Internet. Today, 8.4% of all websites use QUIC [3].

QUIC can be used with different congestion control algorithms [1], [2]. Traditional, loss-based congestion control algorithms like CUBIC [4] treat lost packets as a sign of congestion and consequently reduce the sending rate while retransmitting lost packets. These retransmissions increase the latency and reduce the available bandwidth.

By using Explicit Congestion Notification (ECN), the sender can be notified of congestion before packets have to be dropped. Routers can set a codepoint in the IP header to signalize incipient congestion. When a packet with this codepoint set arrives at the receiver, the receiver has to mirror this information back to the sender. The sender then reduces its sending rate. To achieve this, support of higher layer protocols, such as TCP or QUIC, is necessary. [5]

In the following, we show how ECN can be used with QUIC to optimize QUIC's congestion control. Section 2 provides background information about QUIC and how ECN can be used with TCP. Section 4 focuses on how ECN works with QUIC. In Section 5, we take a look at the support of ECN with QUIC in the Internet. Next, Section 6 discusses an idea of how congestion control with ECN could be further improved. Section 7 concludes this paper by providing a short summary.

2. Background

To understand how ECN with QUIC works, we first take a look at the QUIC protocol itself and how ECN can be used with TCP.

2.1. QUIC

Langley et al. [1] demonstrated their experiences at Google with QUIC in 2017. According to them, using the transport layer protocol TCP comes with various drawbacks. First, using Transport Layer Security (TLS) on top of TCP increases the delay by requiring both a TCP and TLS handshake. Second, multiplexing TCP streams can lead to head-of-line blocking delays.

Making changes to TCP to cope with these challenges is difficult [1]. Since TCP is implemented in the kernel, deploying changes takes time. Creating a completely new transportation protocol on layer 4 is challenging due to middleboxes like Network Address Translations (NATs) or Firewalls, as they would explicitly need to be adapted to support the new protocol. QUIC circumvents this by building on top of the User Datagram Protocol (UDP).

QUIC solves the previously mentioned problems of TCP [1]. To establish a new, secure connection, QUIC only needs a one round-trip time (1-RTT) handshake by combining the TLS and transport layer handshake. Under certain conditions, subsequent connections to the same server can be established using a 0-RTT handshake.

A single QUIC packet can consist of multiple frames, each containing data of a specific stream. Losing a UDP datagram only affects the streams contained in that datagram. Therefore, head-of-line blocking can be avoided. [1]

QUIC does not require a specific congestion control mechanism, making it possible to use different algorithms [1].

2.2. ECN

When using loss-based congestion control, nodes drop packets in case of a full buffer to signalize congestion [5, Section 1], [6]. The sender is able to detect congestion due to duplicate acknowledgments or timeouts and consequently reduces its congestion window [6].

Loss-based congestion control algorithms tend to keep buffers full, leading to a queuing delay [5], [7]. Both full buffers and retransmissions increase the latency. Active Queue Management (AQM) algorithms [8] like Random Early Detection (RED) [9] are trying to avoid the build-up of large queues at routers. While this reduces the queuing

 doi: 10.2313/NET-2025-05-1_13

delay, ECN mitigates the issue of retransmissions due to dropped packets [5].

2.2.1. ECN on the IP layer. ECN makes it possible to inform the sender about congestion without dropping packets. This can be done by setting the Congestion Experienced (CE) codepoint in the IP header [5].
RFC 3168 [5] refers to a packet with the CE codepoint set as a "CE packet". In the following, we use the same definition.

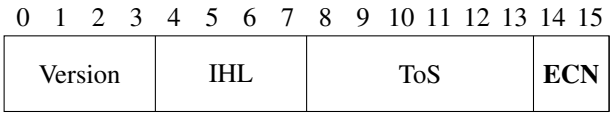

Figure 1: First 2 Byte of the IPv4 header. Based on RFC 791 [10, Figure 4] and RFC 3168 [5, Figure 2]

RFC 3168 defines four ECN codepoints by using bits 6 and 7 of the type of service (ToS) and traffic class field of the IPv4 and IPv6 header, respectively [5, Section 5]. Figure 1 shows the first two bytes of the IPv4 header, including the ECN field. The codepoint "00" is set if ECN is not being used. "11" is the CE codepoint. Either ECT(0) ("01") or ECT(1) ("10") are set by the transport protocol endpoints if they support ECN. Originally, both codepoints were handled equally by routers and served as a one-bit nonce. [5]

More recently, however, ECT(1) serves as a Low Latency Low Loss and Scalable Throughput (L4S) identifier [11]. Routers supporting L4S set the CE codepoint earlier, enabling faster reactions by the endpoints.

2.2.2. Compatibility of ECN. Since the adoption of ECN by routers and hosts happens gradually and is therefore not supported or used by every router and host, it is important that ECN works alongside existing congestion control algorithms. Thus, and to ensure fairness, hosts have to react to an ECN in the exact same way as to a dropped packet. In addition, routers are only allowed to set the CE codepoint if the packet would have been dropped to signal congestion when not using ECN. Routers should deal with a CE packet just as with any other packet. If a queue is entirely full, routers still have to drop incoming packets, even when ECN is used. [5]

2.2.3. ECN with TCP. If an endpoint receives a CE packet, the endpoint has to mirror this information back to the sender so the sending rate can be reduced. To achieve this, ECN requires support from the Transport Layer. [5]

First, when setting up a connection, both endpoints have to be able to signal their capability and willingness to use ECN (Section 2.2.4). Then, after agreeing to use ECN, both endpoints have to react to CE packets by reducing their sending rate (Section 2.2.5). [5]

TCP supports ECN by introducing two flags. Bits 8 and 9 of the reserved field of the TCP header are used for a congestion window reduced (CWR) flag and ECN-Echo (ECE) flag, respectively. Figure 2 shows the CWR and ECE flags. [5, Section 6]

2.2.4. Negotiating the use of ECN. If a host is willing and capable of using ECN, it has to send a packet with

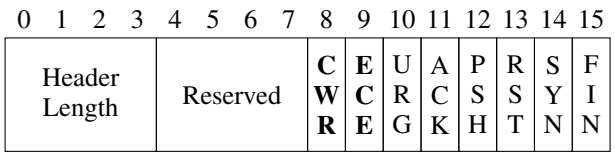

Figure 2: TCP header supporting ECN.
Based on RFC 3168 [5, Figure 4]

the ECE, CWR, and the SYN flag set in the TCP header. The receiver of this packet can then signal their support of ECN by setting the ACK, SYN, and ECE flags but not the CWR flag in the response. This response is then acknowledged by a packet with the ACK flag set. After completion, both endpoints have to react appropriately to CE packets and to TCP segments with the ECE flag set. However, they do not have to set the ECN-capable Transport (ECT) codepoint in the IP header themself. [5]

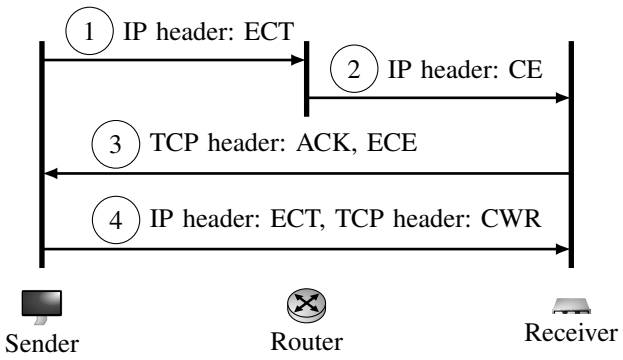

Figure 3: ECN with TCP

2.2.5. Using ECN. Figure 3 illustrates the use of ECN with TCP. When sending a packet, the sender sets the ECT(0) or ECT(1) codepoint in the IP header (1). Then, a router on the network path experiences congestion. Since the ECT codepoint in the IP header of our packet is set, the router can mark the packet with the CE codepoint (2). When receiving a CE packet, the receiver sets ECE in the TCP header (3) to mirror the information back to the sender. Upon receiving a TCP segment with the ECE flag set, the sender reduces its congestion window and informs the receiver about that by setting the CWR flag (4). As soon as the receiver processes this flag, it stops setting ECE in the TCP header. [5]

3. Related work

Most of the related, previous work has focused on ECN with TCP. However, more recent work has also studied improvements and the support of using ECN with QUIC.

ECN with TCP. Floyd [12] has conducted simulations to point out several advantages of using ECN with TCP. For instance, in one of their LAN simulation scenarios, they were using ten telnet connections, a 0.1 msec TCP clock, packet-based RED gateways (with and without ECN), and a 64 kB maximum TCP window. In this experiment, they were able to decrease the average delay from about 20ms to nearly zero by using ECN.

 doi: 10.2313/NET-2025-05-1_13

Different simulation scenarios show similar results: by using ECN, fewer packets were dropped, which decreased delays. They also pointed out potential disadvantages, mainly focusing on misbehaving endpoints and losing acknowledgment (ACK) packets.

A variety of other studies have focused on the support of ECN on the Internet. Lim et al. [13] have shown in 2022 that over 85% of Alexa Top 100K web servers support ECN with TCP. Bauer et al. [14] have observed a similar number while conducting Internet measurements regarding the effect of TCP options. This is a massive increase from what previous work has shown earlier. For instance, Bauer et al. [15] conducted experiments using the Alexa Top 1M list in 2011. Only about 17% of web servers supported ECN at that time.

ECN with QUIC. While ECN with TCP is widely deployed in today's Internet, Sander et al. [16] were able to show that the opposite holds for QUIC. By conducting extensive research and experiments on the support of ECN with QUIC in the Internet, they were able to demonstrate that by the time of their studies, ECN could be used with less than 2% of QUIC hosts. We take a closer look at their results in Section 5. Uchida et al. [17] proposed a method to leverage ECN with QUIC to improve the fairness of two competing hosts using QUIC with CUBIC and BBR [7], respectively. By adapting how CUBIC reacts to ECN codepoints and by adapting BBR's phase transitions, they were able to improve fairness in their experiments. For instance, when using a bottleneck link buffer of 1 Mbit, they improved Jain's fairness index value from less than 0.7 to nearly 1.

In this paper, we mainly focus on how ECN with QUIC works (Section 4) and what current impediments toward a wide deployment of ECN with QUIC are (Section 5).

4. ECN with QUIC

This section is based on RFC 9000 [2], which contains specifications on how ECN can be used with QUIC. Similar to using ECN with TCP, the sender decreases its sending rate when receiving a CE packet. In contrast to using ECN with TCP, ECN with QUIC can also be used in only one direction. In addition, the receiver informs the sender not only about CE packets but also about the ECT(0) and ECT(1) codepoints it receives. In order to use ECN with QUIC, a sender first has to confirm that both the receiver and intermediate nodes support ECN.

4.1. Mirroring ECN Counts

To be able to use ECN with QUIC, the receiver needs to be able to access bits 6 and 7 of the ToS and traffic class field of the IPv4 and IPv6 header, respectively. The receiver then maintains counts for the ECT(0), ECT(1), and CE codepoints it has observed. These counts can be mirrored back to the sender using specific fields in the ACK frames.

Similar to TCP, ACK frames in QUIC are used to confirm successfully transmitted packets [2, Section 19.3]. To support ECN, QUIC introduced the ACK frame type 0x03, as shown in Figure 4. ACK frames of this type additionally include the ECN counts of the receiver for the packet number space it acknowledges. By using this ACK frame type, the receiver is able to inform the sender about the total number of ECT(0), ECT(1), and CE codepoints it received.

```
ACK Frame {
    Type = 0x03,
    Largest Acknowledged,
    ACK Delay,
    ACK Range Count,
    First ACK Range,
    ACK Range ...,
    ECN Counts {
        ECT0 Count,
        ECT1 Count,
        ECN-CE Count,
    }
}
```

Figure 4: ACK frame format of type 0x03 in QUIC. Based on RFC 9000 [2, Figure 25 and Figure 27]

4.2. Example: ECN with QUIC

Figure 5 illustrates how QUIC with ECN works. After establishing a connection and agreeing on the use of ECN, the sender sets the ECT(0)/ ECT(1) codepoint in the IP header (1) to signal the use of ECN to routers. In (2), the router experiences a congestion. Instead of dropping the packet, it sets the CE codepoint in the IP header (3). The receiver maintains the ECN counts n, m, and k for the number of ECT(0), ECT(1), and CE codepoints received (4). The receiver includes these counts in the ACK frame (5, 6) when acknowledging the received packet. Due to the increase in the CE count, the sender then reduces its sending rate.

4.3. ECN validation

To determine whether the network path supports ECN, the sender sets ECT(0) (or ECT(1)) in the first few packets. If none of these packets are acknowledged, the sender assumes the packets have been dropped and that the path does not support ECN. The sender then disables ECN.

If the sender receives an ACK frame containing the ECN counts, he has to validate them [2, Section 13.4]. It is important that all ACK frames being used for ECN validation increase the largest acknowledged packet number.

There are several scenarios that lead to a failed ECN validation. In the following, we define *A_ECT_0* as the ECT(0) count in the current ACK frame. *S_ECT_0* stands for the total number of ECT(0) codepoints set by the sender. Similarly, Δ *A_ECT_0* defines the difference between the ECT(0) count in the current ACK frame and the ECT(0) count in the previous ACK frame. Δ *S_ECT_0* represents the number of packets that are newly acknowledged and were sent with the ECT(0) codepoint set. In the same way, *A_ECT_1*, Δ *A_ECT_1*, Δ *A_CE* and *S_ECT_1* are defined.

As specified in [2, Section 13.4], the following conditions are verified:

 doi: 10.2313/NET-2025-05-1_13

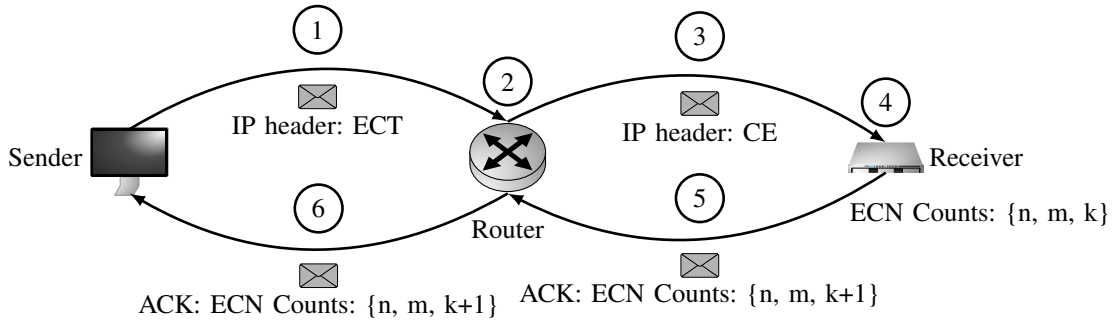

Figure 5: ECN with QUIC

1) The ACK frame contains ECN counts
2) ΔA_ECT_0 + ΔA_ECT_CE $\geq$ ΔS_ECT_0
3) ΔA_ECT_1 + ΔA_ECT_CE $\geq$ ΔS_ECT_1
4) A_ECT_0 $\leq$ S_ECT_0
5) A_ECT_1 $\leq$ S_ECT_1

If any of these conditions do not hold, ECN has to be deactivated. 1 validates that the receiver mirrors the ECN codepoints and that the ECN field of the IP header does not get cleared from a node on the network path. 2 and 3 are used to detect the remarking of ECN codepoints. For instance, a node on the network path could change a CE codepoint to ECT(0), although the packet was initially sent with an ECT(1) codepoint. Since ACK frames can get lost, it is possible that e.g. ΔA_ECT_0 >= ΔS_ECT_0 holds, which explains the inequality used in 2 and 3. However, the total number of A_ECT_0 / A_ECT_1 can never exceed S_ECT_0 / S_ECT_1. Thus, 4 and 5 are also used to detect the remarking of ECN codepoints.

5. Support of ECN with QUIC

One of the biggest challenges of using ECN over QUIC is the lack of support. Sander et al. [16] have been able to show that by the time of their study in 2023, QUIC with ECN could only be used with less than 2% of the hosts they investigated. In order to use ECN with QUIC, the codepoints have to be mirrored (see Section 4.1), and the validation of the ECN codepoints has to succeed (see Section 4.3). By conducting several experiments, they were able to pinpoint the causes of the low support of ECN with QUIC. This section discusses the main findings of Sander et al. [16].

5.1. Missing ECN counts

In case it is possible to access the ECN codepoints of the IP header, the QUIC standard [2, Section 13.4] defines the mirroring of ECN codepoints as a MUST. However, a previous statement in the standard makes QUIC support seem to be optional, causing ambiguities [18]. Actually, only 20% of QUIC hosts that were tested by Sander et al. include the ECN counts in the ACK frames. The interop runner [19] shows that only 6 out of 17 tested QUIC implementations support ECN.

5.2. Undercounting of ECN codepoints

Mirroring the ECN codepoints is not sufficient for using ECN with QUIC. Instead, the sender also validates the reported ECN counts (see Section 4.3). Sander et al. [16] were able to show that over 90% of the domains that support ECN mirroring fail this validation stage. Over one-half of them acknowledged fewer ECN codepoints than they had been sent with. They were able to show that this is mainly because of issues in the QUIC implementation used by the receiver and not due to nodes in the network.

5.3. Remarking of ECN codepoints

About one-third of the domains investigated report ECT(0) codepoints as ECT(1) codepoints. This is, however, not caused by the used QUIC stack but mainly by the network operator Arelion (ASN 1299). When repeating the measurements from various geographically distributed origins, the overall observed pattern stays the same. In fact, globally, only about 0.3% of the tested domains meet all the requirements during the validation phase. Even if the validation is successful, it does not mean that the endpoint or routers on the network path actually use ECN. [16]

6. Possible Improvement

The current QUIC standard [2] specifies the use of ACK frames with the type 0x03 when using ECN. As explained in Section 4.1, these ACK frames contain the total count of ECN codepoints the receiver has observed. This information, for instance, is used by the Prague Congestion Control Algorithm [20]. However, it still lacks the information on which packet was marked with which codepoint. This fine-grained information could be valuable to react even more efficiently and effectively to CE packets, according to Seemann et al. [21]

Therefore, they proposed a new QUIC ACK frame type, which includes the ECN codepoint that was set in the packets of each ACK range. If necessary, ranges have to be split into multiple ranges such that all packets within a range share the same ECN codepoint. [21]

It remains to be seen if this idea will be included in future versions of QUIC and if congestion algorithms can actually profit from this fine-grained information.

7. Conclusion

In this paper, we have explained how ECN works and how it can be used with QUIC. ECN is used to notify a

sender about congestion in the network. To achieve this, a router can mark a packet with the congestion experienced codepoint. QUIC then mirrors the codepoint back to the sender using ACK frames, and the sender can reduce its sending rate.

We have also discussed that the biggest impediment of using ECN with QUIC is the missing support in common QUIC implementations and misbehaving network operators.

While ECN can optimize QUIC's congestion control by trying to avoid retransmissions, it is currently barely used. Having more QUIC implementations supporting ECN would be essential for a wide adoption and usage of ECN with QUIC. In order to demonstrate the impact of using ECN with QUIC on throughput, latency, and packet drops quantitatively, performance measurements and comparisons could be conducted as part of future work. Depending on the results, this could speed up the support and use of ECN with QUIC.

References

[1] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 183–196. [Online]. Available: https://doi.org/10.1145/3098822.3098842

[2] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[3] W3Techs, "Usage statistics of QUIC for websites," https://w3techs.com/technologies/details/ce-quic, [Online; accessed 29-November-2024].

[4] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks," RFC 8312, Feb. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8312

[5] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, Sep. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3168

[6] E. Blanton, D. V. Paxson, and M. Allman, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[7] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, p. 20–53, Oct. 2016. [Online]. Available: https://doi.org/10.1145/3012426.3022184

[8] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567, Jul. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7567

[9] L. Zhang, D. C. Partridge, S. Shenker, J. T. Wroclawski, D. K. K. Ramakrishnan, L. Peterson, D. D. D. Clark, G. Minshall, J. Crowcroft, R. T. Braden, D. S. E. Deering, S. Floyd, D. B. S. Davie, V. Jacobson, and D. D. Estrin, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Apr. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2309

[10] J. Postel, "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[11] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," RFC 9330, Jan. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9330

[12] S. Floyd, "TCP and explicit congestion notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, p. 8–23, Oct. 1994. [Online]. Available: https://doi.org/10.1145/205511.205512

[13] H. Lim, S. Kim, J. Sippe, J. Kim, G. White, C.-H. Lee, E. Wustrow, K. Lee, D. Grunwald, and S. Ha, "A Fresh Look at ECN Traversal in the Wild," *arXiv preprint arXiv:2208.14523*, 2022.

[14] S. Bauer, P. Sattler, J. Zirngibl, C. Schwarzenberg, and G. Carle, "Evaluating the Benefits: Quantifying the Effects of TCP Options, QUIC, and CDNs on Throughput," in *Proceedings of the 2023 Applied Networking Research Workshop*, ser. ANRW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 27–33. [Online]. Available: https://doi.org/10.1145/3606464.3606474

[15] S. Bauer, R. Beverly, and A. Berger, "Measuring the state of ECN readiness in servers, clients, and routers," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 171–180.

[16] C. Sander, I. Kunze, L. Blöcher, M. Kosek, and K. Wehrle, "ECN with QUIC: Challenges in the Wild," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, 2023, pp. 540–553.

[17] N. Uchida, D. Nobayashi, D. Cavendish, and T. Ikenaga, "Poster: Fairness improvement method using ECNs with different congestion control algorithms within QUIC," in *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. IEEE, 2023, pp. 1–2.

[18] "Technical Errata Reported RFC9000 ECN," https://mailarchive.ietf.org/arch/msg/quic/lsz4X-cZql71Ba56uQhNQz4NzGc/, 2023, [Online; accessed 28-November-2024].

[19] M. Seemann, "QUIC Interop runner," https://interop.seemann.io/, [Online; accessed 01-March-2025].

[20] K. D. Schepper, O. Tilmans, B. Briscoe, and V. Goel, "Prague Congestion Control," Internet Engineering Task Force, Internet-Draft draft-briscoe-iccrg-prague-congestion-control-04, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-briscoe-iccrg-prague-congestion-control/04/

[21] M. Seemann and V. Goel, "QUIC Accurate ECN Acknowledgements," Internet Engineering Task Force, Internet-Draft draft-seemann-quic-accurate-ack-ecn-01, May 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-seemann-quic-accurate-ack-ecn/01/