Quic meets ICN: A versatile wireless transport strategy in multi-access edge environments

Abstract

In the previous work, we built up a system that can be nearly seamless using QUIC-based applications on the information-centric networking (ICN) wireless communication architecture and tested the performance and applicability of the system, ICN-QUIC. For a further investigation of ICN-QUIC, we propose the possibility of how ICN-QUIC can improve its communication performance. We also complete the vanilla QUIC functionality of ICN-QUIC and envision practical scenarios that can better leverage the ICN advantages. We show the related work and possible mechanisms in this report and will finish them in the next work.

Keywords: Information-centric Networking (ICN), QUIC, Congestion Control, Reliability.

1 Introduction

ICN-QUIC system is actually a protocol stack designed to improve data transmission in wireless multi-access edge computing environments [8]. It combines the strengths of ICN with the QUIC transport protocol, making data sharing more efficient and adaptable for edge networks. ICN-QUIC enables efficient data transmission while maintaining compatibility with existing QUIC-based applications. In detail, the system introduces an ICN-QUIC conversion layer that modifies QUIC for ICN links, allowing seamless communication between devices using ICN without modifying userspace applications. It also incorporates a new transport layer protocol called QVDP (QUIC-supported Versatile Datagram Protocol) that supports efficient packet delivery across different layers. ICN-QUIC supports both unicast and multicast communication, leveraging the ability of ICN to handle data names instead of traditional IP addresses.

According to the above, we can conclude that QUIC had been used for the extension of the ICN application base, including unicast and multicast scenarios. The basic function of a few QUIC applications and scalability of ICN-QUIC had also been tested. ICN-QUIC had mentioned that userspace implementation of QUIC and custom extended frame can be some reasons for using QUIC over ICN.

However, we still need to think about the contrary problem: What does QUIC benefit from ICN mode? There are two reasons for addressing this problem. First, the final form for the user of ICN-QUIC is the QUIC application, figuring out this problem can help to design suitable QUIC applications in some ICN-specific scenarios to achieve a better performance. A new study also points out the possible performance defects of QUIC [21]. Second, many mechanisms of QUIC, especially performance-related, still need to be adapted for the ICN mode for a deep vision of ICN-QUIC and a non-decreased throughput. For example, the flow and congestion control in two scenarios, 0-RTT ability, and Connection Migration. We would be better to carefully quantify these adaptations to indicate the performance upper and lower limits of ICN-QUIC.

In this report, we focus on two main tasks. First, we extend the experiments from the original project by adding new tests related to scalability and exploring the impact of different hyperparameters. Second, we look into ways to improve the ICN-QUIC system. Specifically, we explore how to better leverage the strengths of QUIC in different scenarios and how to seamlessly adapt the transmission mechanisms of QUIC (e.g., flow control and congestion control) to fit the needs of the ICN-QUIC framework. The goal is to enhance the overall performance and make the system more efficient without causing disruption to the existing setup.

2 Related works

2.1 QUIC transmission mechanism

Recently, many works have studied the congestion control [10], flow control, loss recovery, and acknowledgment methods [9] of QUIC. Custura *et al.* look at how different acknowledgment policies in QUIC can reduce traffic and processing costs without hurting performance, especially in satellite, cellular, and terrestrial networks [3]. Liu *et al.* propose an ACK frequency optimization scheme, QUIC-BDP, for Wi-Fi-enabled IoT communications, improving QUIC performance [13]. Mucke *et al.* [15] analyze the performance of QUIC instant ACK, showing that while it improves connection setup times in CDN deployments, it can also cause unnecessary retransmissions or longer delays under certain network conditions. Xiao *et al.* [19] introduce the CubicBytes-N algorithm, an improved congestion control mechanism for QUIC in wireless networks, which better distinguishes between congestion and random packet loss, especially in high packet loss environments. Zhang *et al.* [22] propose PBQ-enhanced QUIC which combines BBR and PPO to improve throughput and reduce RTT compared to traditional QUIC versions like those using Cubic or BBR. Chen *et al.* [1] propose a reinforcement learning-based congestion control algorithm for multicast QUIC, using Decaying epsilon-greedy and critic-actor methods, which significantly reduces packet loss and adapts well to dynamic network environments. Fernandez *et al.* [5, 6] present a priority-based stream scheduler for QUIC, designed to improve performance in mmWave networks, showing that it can reduce delays for time-sensitive applications under unreliable conditions. Michel *et al.* [14] introduce QUIRL, a new loss recovery mechanism for QUIC that uses Forward Erasure Correction only when necessary, showing significant performance improvements in real-time video and HTTP/3 applications over lossy networks. He *et al.* [7] present ShuttleBus, a dense packet assembling scheme for mIoT, which uses QUIC stream multiplexing to reduce communication overhead while meeting strict latency constraints.

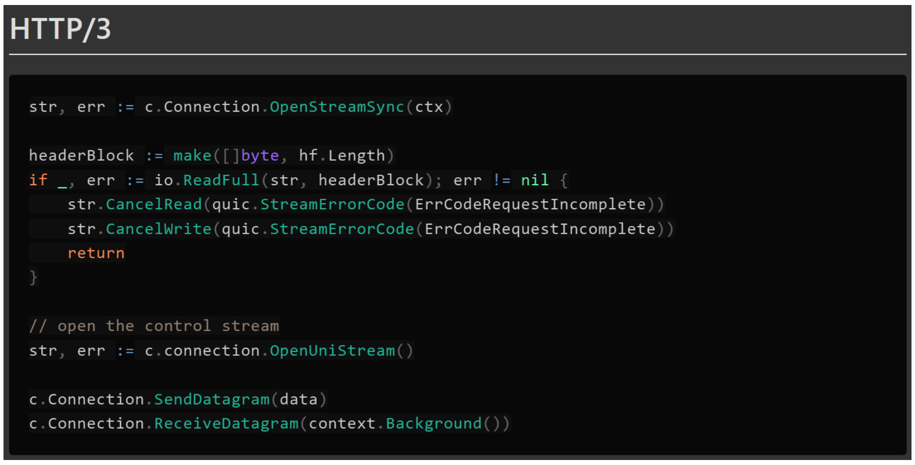
2.2 Advantageous scenarios for QUIC

QUIC has been tested on the internet, but its performance has not yet met expectations [21]. As a result, many studies are now exploring in which scenarios the complex designs and mechanisms of QUIC can provide advantages. Tauqeer *et al.* [17] present Q-MOFI, a mobile-friendly internet architecture based on QUIC, showing a better result of throughput, reduces packet loss, and lowers latency compared to traditional UDP-based MOFI. Jaeger *et al.* [11] extend the QUIC Interop Runner to enable reproducible benchmarks on dedicated hardware, evaluating QUIC performance across different implementations and hardware configurations, showing significant variations in throughput and highlighting the impact of OS buffer sizes and NIC offloading. Kim *et al.* [12] introduce mQUIC using QUIC connection migration to support seamless handovers for mobile clients in wireless networks, demonstrating significant performance improvements. Wang *et al.* [18] introduce CoMPS that leverages connection migration features to split traffic across network paths, enhancing user privacy by mitigating traffic analysis attacks. Conforti *et al.* [2] extend QUIC to support server-side connection migration during container migrations, proposing two strategies and testing them with aioquic. Yan *et al.* [20] explore the feasibility of using QUIC connection migration in a multi-carrier Wi-Fi relay to reduce latency caused by cellular suspension.

3 Method

3.1 Application Investigation

To enhance the application compatibility of ICN-QUIC, we examine the source code of existing open-source QUIC-based projects and find that these applications typically use a unified interface to call QUIC, mainly for stream opening, writing, and reading operations, as shown in Figure 1. However, these interfaces may be invoked multiple times during the application lifecycle. Therefore, adapting the Interest-Data mechanism of ICN to this process presents a challenge, while still maintaining the fundamental characteristics of full-duplex communication.


HTTP/3

```
str, err := c.Connection.OpenStreamSync(ctx)

headerBlock := make([]byte, hf.Length)
if _, err := io.ReadFull(str, headerBlock); err != nil {
    str.CancelRead(quic.StreamErrorCode(ErrCodeRequestIncomplete))
    str.CancelWrite(quic.StreamErrorCode(ErrCodeRequestIncomplete))
    return
}

// open the control stream
str, err := c.connection.OpenUniStream()

c.Connection.SendDatagram(data)
c.Connection.ReceiveDatagram(context.Background())
```

Figure 1. Code example of QUIC application

3.2 Advantage scenario of ICN-QUIC

We have carefully reconsidered the 0-RTT and connection migration features of QUIC. First, the ICN radio [4] in ICN-QUIC is a communication model that does not require maintaining persistent connections, making it well-suited for mobile scenarios. Secondly, both 0-RTT and connection migration are likely to occur in situations where network conditions change frequently. Specifically, 0-RTT is better suited for environments where connections are frequently established and torn down, such as in energy-constrained IoT communications. On the other hand, connection migration is triggered by real network changes, like switching to a different network interface.

Given these points, it is clear that ICN-QUIC has significant potential to efficiently adapt both 0-RTT and connection migration for reliable data communication in mobile scenarios involving wireless user terminals. We envision a scenario where a cluster of IoT edge servers is deployed, and multiple mobile users move around, for example, transitioning from the coverage area of server A to that of server B, while sharing data among themselves. In such a scenario, ICN-QUIC would need to adapt its 0-RTT and connection migration features to maintain seamless communication. For 0-RTT, ICN-QUIC needs to develop intelligent caching mechanisms that can quickly validate and reuse session information without compromising data integrity. For connection migration, ICN-QUIC would need to enhance its ability to detect network changes rapidly and switch contexts seamlessly between different network interfaces. This could involve developing more sophisticated algorithms for state synchronization and ensuring that data packets are efficiently rerouted without significant delays or data loss.

3.3 Transmission control

We are still in the process of designing optimizations for congestion control, flow control, and reliable transmission. We outline some potential directions for solutions. First, the basic mechanism of flow control, due to its simplicity, could be retained, but it may require a selection mechanism to determine the updated value of the flow control window and a stronger guarantee mechanism to ensure that flow control frames reach the destination when approaching the window limit. Second, regarding congestion control algorithms, due to their diversity and the goal of addressing congestion issues in multi-hop networks, we may need to discard previous congestion control algorithms and design one that mitigates traffic conflicts caused by multiple terminals in a single-hop environment. Additionally, these mechanisms need to consider multicast, not just unicast.

4 Implementation details

4.1 Comparing with the released source codes

We refer to the source code of ICN-QUIC, which is developed based on quic-go [16]. We made corresponding modifications in the latest version of quic-go and conducted some tests.

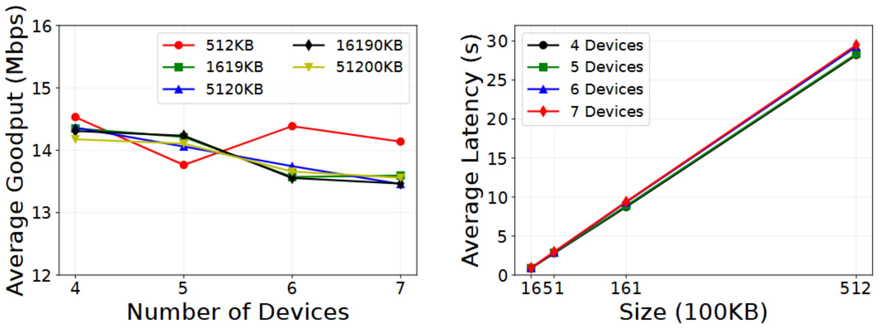
4.2 Experimental environment setup

We use several Raspberry Pi 4B and Raspberry Pi 5 with commodity-available USB Wi-Fi devices using Atheros AR9271 chip sets, each with one 6 dBi antenna. The frequency bandwidth of the dongles is 20 MHz, and the theoretical maximum wireless speed is 18.75 MB/s. We use 8 Pis, in which 1 serves as the publisher/sender and the other 7 serve as subscribers/receivers. In the scalability evaluation, we use files ranging from 512 KB to 51200 KB in size.

5 Results and analysis

Figure 2 (a) illustrates the average goodput of all devices of varying sizes across different numbers of devices. The trend of the average goodput is essentially consistent when requesting 1619 KB to 51200 KB data files, with slight fluctuations observed in the 512 KB data request tests. The greater fluctuations for the 512KB evaluation are due to the variability inherent in wireless networks, which has a more pronounced effect on smaller data sizes. In the four rounds of tests with different numbers of devices, the variances in goodput for obtaining files of different sizes are 0.013, 0.028, 0.096, and 0.064, respectively. This shows that ICN-QUIC system is relatively stable in supporting large files. We also observe that as the number of devices increases, the average goodput decreases from 14.53 to 13.46 Mbps, which decreases by 7.36 %. This is because each device may experience packet loss, and as the number of devices increases, the overall packet loss rate rises. Consequently, the retransmission mechanism of the ICN-QUIC system needs to send more packets for reliability, thus slightly increasing the overall latency of the system. Although there is a slight decrease in goodput, the extent is not significant.

Figure 2 (b) illustrates the increase in average latency time for all devices to complete the data requests as both the file size and the number of devices increases. We notice that as the number of devices making data requests increases from 4 to 7, the time required for data transfer increases in a linear manner, ranging from 1619 KB to 51200 KB. This shows that in ICN-QUIC system, the latency required to complete data transmission is only related to the size of the file and not to the number of devices, allowing for stable transmission even with a larger number of devices. In conclusion, the above two experiments demonstrate that ICN-QUIC system is able to stably support not only a higher number of devices but also larger files in transmission.



(a) Average goodput (b) Average latency

Figure 2. The performance over multiple devices and larger files for (a) average goodput and (b) average latency of the system

To assess the stability of the loss recovery mechanism introduced by the ICN-QUIC system, we conduct experiments that focus on quantifying the cost of retransmission requests during multiple rounds of transmission. All the experiment we conduct in this part adopts reliable transmission.

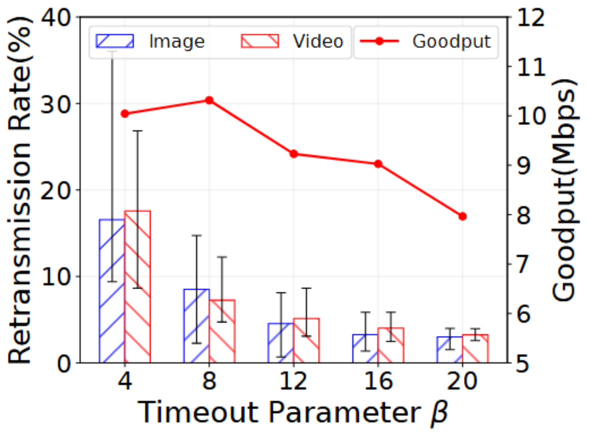


Figure 3. The retransmission rate and goodput for different files and different parameter settings of β

Figure 3 denotes the retransmission rate and average goodput in different settings of β. It is obvious that when the β is larger, which means the timeout is longer and we will wait longer for the packet to arrive, the rate of retransmission is lower. On average, when β is set to 20, the retransmission rate is only about a quarter of that when $\beta = 4$. On one hand, if the timeout is short, more retransmission will cause more contention, which will have a negative impact on the transmission rate. On the other hand, longer timeout also increases the waiting time to get the data, also decreasing the goodput. Thus, the trade-off needs to be considered to balance retransmission and goodput. Fig. 3 also shows that when the parameter β is set to 8, the system achieves the largest goodput with an average of 10.44 Mbps.

6 Conclusion and future work

In this report, we have mainly described how ICN-QUIC performs performance optimization and adapts to different scenarios. We highlight the issues and challenges that need to be addressed, along with some preliminary directions for potential solutions. We have not yet designed specific algorithms and mechanisms, which will be part of our future work. Our ultimate goal is to build up a protocol that supports a seamless transfer between UDP/IP and ICN for the applications over QUIC and provides efficient data sharing, especially in multicast and mobility edge scenarios.

References

[1] Yutong Chen, Haoran Shi, Qiyan Weng, and Zhiguo Shi. Congestion control design of multicast quic based on reinforcement learning. In *2023 International Conference on Ubiquitous Communication (Ucom)*, pages 232–236. IEEE, 2023.

[2] Luca Conforti, Antonio Virdis, Carlo Puliafito, and Enzo Mingozzi. Extending the quic protocol to support live container migration at the edge. In *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 61–70. IEEE, 2021.

[3] Ana Custura, Tom Jones, Raffaello Secchi, and Gorry Fairhurst. Reducing the acknowledgement frequency in ietf quic. *International Journal of Satellite Communications and Networking*, 41(4):315–330, 2023.

[4] Mohammed Elbadry, Fan Ye, Peter Milder, and Yuanyuan Yang. Pub/sub in the air: A novel data-centric radio supporting robust multicast in edge environments. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 257–270. IEEE, 2020.

[5] Fátima Fernández, Fátima Khan, Mihail Zverev, Luis Diez, José R Juárez, Anna Brunstrom, and Ramón Agüero. Exploiting stream scheduling in quic: Performance assessment over wireless connectivity scenarios. *Ad Hoc Networks*, 164:103599, 2024.

[6] Fátima Fernández, Mihail Zverev, Luis Diez, José R Juárez, Anna Brunstrom, and Ramón Agüero. Flexible priority-based stream schedulers in quic. In *Proceedings of the Int'l ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pages 91–98, 2023.

[7] Bo He, Jingyu Wang, Qi Qi, Qiang Ye, Qihao Li, Jianxin Liao, and Xuemin Shen. Shuttlebus: Dense packet assembling with quic stream multiplexing for massive iot. *IEEE Transactions on Mobile Computing*, 23(8):8307–8322, 2023.

[8] Yaodong Huang, Changkang Mo, Tianhang Liu, Biying Kong, Lei Zhang, Yukun Yuan, and Laizhong Cui. Quic meets icn: A versatile wireless transport strategy in multi-access edge environments. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE, 2024.

[9] J. Iyengar, I. Swett, and M. Kühlewind. Draft-ietf-quic-ack-frequency-08. `https://www.ietf.org/archive/id/draft-ietf-quic-ack-frequency-08.html`, 2024.

[10] Jana Iyengar and Ian Swett. Rfc 9002: Quic loss detection and congestion control, 2021.

[11] Benedikt Jaeger, Johannes Zirngibl, Marcel Kempf, Kevin Ploch, and Georg Carle. Quic on the highway: evaluating performance on high-rate links. In *2023 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2023.

[12] So-Yong Kim and Seok-Joo Koh. mquic: Use of quic for handover support with connection migration in wireless/mobile networks. *IEEE Communications Magazine*, 2023.

[13] Yang Liu, Zhaoxian Yang, Yuxiang Peng, Ting Bi, and Tao Jiang. Bandwidth-delay-product-based ack optimization strategy for quic in wi-fi networks. *IEEE Internet of Things Journal*, 10(20):17635–17646, 2023.

[14] François Michel and Olivier Bonaventure. Quirl: Flexible quic loss recovery for low latency applications. *IEEE/ACM Transactions on Networking*, 2024.

[15] Jonas Mücke, Marcin Nawrocki, Raphael Hiesgen, Thomas C Schmidt, and Matthias Wählisch. Reacked quicer: Measuring the performance of instant acknowledgments in quic handshakes. In *Proceedings of the 2024 ACM on Internet Measurement Conference*, pages 389–400, 2024.

[16] Marten Seemann. quic-go. https://github.com/quic-go/quic-go, 2025.

[17] Muhammad Tauqeer, Moneeb Gohar, Seok Joo Koh, and Hani Alquhayz. Use of quic for mobile-oriented future internet (q-mofi). *Electronics*, 13(2):431, 2024.

[18] Mona Wang, Anunay Kulshrestha, Liang Wang, and Prateek Mittal. Leveraging strategic connection migration-powered traffic splitting for privacy. *arXiv preprint arXiv:2205.03326*, 2022.

[19] Xiang Xiao, Long Chen, and Ming Zhao. Optimized congestion control algorithm for quic in wireless networks: Cubicbytes-n algorithm. *IEEE Internet of Things Journal*, 2024.

[20] Yihui Yan and Zhice Yang. When quic's connection migration meets middleboxes a case study on mobile wi-fi hotspot. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[21] Xumiao Zhang, Shuowei Jin, Yi He, Ahmad Hassan, Z Morley Mao, Feng Qian, and Zhi-Li Zhang. Quic is not quick enough over fast internet. In *Proceedings of the ACM on Web Conference 2024*, pages 2713–2722, 2024.

[22] Zhifei Zhang, Shuo Li, Yiyang Ge, Ge Xiong, Yu Zhang, and Ke Xiong. Pbq-enhanced quic: Quic with deep reinforcement learning congestion control mechanism. *Entropy*, 25(2):294, 2023.