RFC 9000 and its Siblings: An Overview of QUIC Standards

YaXuan Chen, Benedikt Jaeger*, Johannes Zirngibl*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: chenyaxuan123@gmail.com, jaeger@net.in.tum.de, zirngibl@net.in.tum.de

***Abstract*—QUIC was a transport protocol designed by Google in 2012, aiming to improve traditional transport protocols, such as TCP. QUIC reduces the handshake latency and solves the head-of-line blocking. The IETF (Internet Engineering Task Force) standardized QUIC in 2021 and continues to improve and extend the QUIC transport protocol. There are 12 RFCs (Request for Comments) related to QUIC, which involve the core concepts of QUIC, QUIC's extensions, applicability, manageability, and HTTP/3. In this paper, we describe the core concept of QUIC based on RFC 9000 and outline four more RFCs to provide an overview of QUIC standards. Furthermore, we discuss which direction the QUIC working group plans for the future.**

***Index Terms*—Transport protocols; QUIC; HTTP/3;**

1. Introduction

With the gradual popularization of the Internet, there has been increasingly more attention on high-speed and stable connections. TCP (Transmission Control Protocol) is the first choice for the developer. However, applications are limited by using TCP as the underlying transport [1]. TCP needs 3 round-trip time (RTT) for setting up the first connection from a client to the server. Furthermore, the head-of-line blocking increases the connection latency when a packet losses [2]. Moreover, TCP is commonly implemented in the operating system kernel. Therefore, the TCP modifications require OS upgrades [3].

QUIC is a new transport protocol designed by Google in 2012 to replace the traditional HTTPS stack: HTTP/2, TLS, and TCP [3], as shown in Figure 1. Notably, QUIC builds on top of UDP and uses a different handshake mechanism from TCP [2], which combines the transport and cryptographic handshake and supports the 0-RTT connection. Furthermore, QUIC solves the head-of-line blocking using a lightweight data-structuring abstraction, streams [3], present in Section 3.1. Additionally, QUIC is implemented in user space, which can be updated without changing the operating system kernel. Applications like Facebook, YouTube, and Gmail have supported QUIC. QUIC is used by 7.6% of all the websites in November 2023 [4]. The proportion of QUIC will continue to increase in the future.

The IETF QUIC Working Group [5] is responsible for the maintenance and evolution of QUIC. The first version of QUIC was standardized in May 2021 as RFC 9000 [6]. There are 12 RFCs related to QUIC until May 2023, as shown in Table 1. However, the content of all RFCs is too much, and we decided to describe part of them in this paper, marked in red in Table 1.

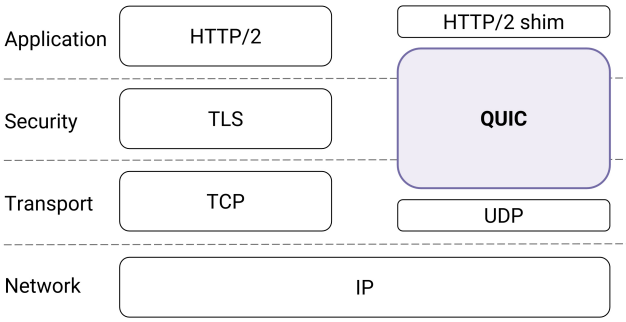

Figure 1: QUIC in the traditional HTTPS stack [3].

The paper is structured as follows: Section 3 describes the core concepts of QUIC. In Section 4, we present an overview of QUIC extensions including RFC 9221 [7], RFC 9368 [8], and RFC 9369 [9]. Furthermore, we present RFC 9114 [10] about HTTP/3. In Section 5, we discuss the future directions of QUIC.

2. Related Work

QUIC related RFCs. This paper provides an overview of QUIC standards and is based on the 12 RFCs [6]–[17].

Core concept of QUIC. Langley et al. in [3] explain the main features of QUIC, and Cui et al. in [1] analyze the advantage of QUIC compared to TCP. Section 3 are closely related to these works.

3. Core Concepts of QUIC

In this section, we outline the core specifications of QUIC from RFC 9000 [6], including QUIC streams, connection migration, and flow control. In addition, we explain how QUIC solves head-of-line blocking. Moreover, we present the congestion control of QUIC from RFC 9002 [7]. Furthermore, we describe the process of 0-RTT handshake from RFC 9001 [12].

3.1. QUIC Streams

QUIC streams are a lightweight abstraction that provides a reliable bidirectional bytestream [3]. Streams consist of multiple stream frames encapsulated in a QUIC packet, as shown in Figure 2. There are two important values in a stream frame:

 doi: 10.2313/NET-2024-04-1_02

TABLE 1: Overview of QUIC-related RFCs

RFCs	Titel	Publication date	Content
8999	Version-Independent Properties of QUIC	May 2021	Definition of the properties of the QUIC transport protocol.
9000	QUIC: A UDP-Based Multiplexed and Secure Transport	May 2021	Description of the core QUIC protocol.
9001	Using TLS to Secure QUIC	May 2021	Functionalities of TLS in QUIC.
9002	QUIC Loss Detection and Congestion Control	May 2021	Loss detection and congestion control mechanisms for QUIC.
9221	An Unreliable Datagram Extension to QUIC	March 2022	QUIC Extension, which supports unreliable datagrams.
9287	Greasing the QUIC Bit	August 2022	Definition of the usage of QUIC Bit.
9368	Compatible Version Negotiation for QUIC	May 2023	The update of the version negotiation mechanisms for QUIC.
9369	QUIC Version 2	May 2023	Definition of the second version of QUIC.
9308	Applicability of the QUIC Transport Protocol	September 2022	Caveats for the developers who want to use QUIC.
9312	Manageability of the QUIC Transport Protocol	September 2022	Guidance for network operations to manage QUIC traffic.
9114	HTTP/3	June 2022	The version of HTTP, which uses QUIC as the transport protocol.
9204	QPACK: Field Compression for HTTP/3	June 2022	The mechanismus of QPACK compression.

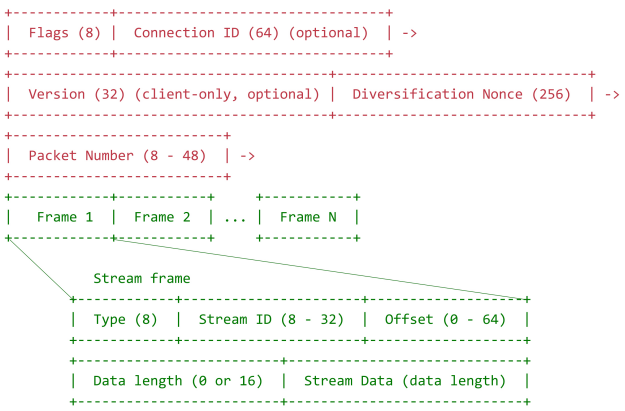

Figure 2: Structure of a QUIC packet [3].

- **Stream ID** is a unique 62-bit integer that identifies the stream. The odd IDs represent client-initiated streams and even IDs for server-initiated streams.
- **Offset** represents in which position the transmitted data should be placed. QUIC packets are encapsulated in UDP packets, and UDP does not transmit data in order. The unordered stream data can be placed in the correct stream and position with stream ID and offset.

Additionally, QUIC also provides unidirectional streams for different purposes (e.g. the control stream in HTTP/3, which is introduced in Section 4.4.)

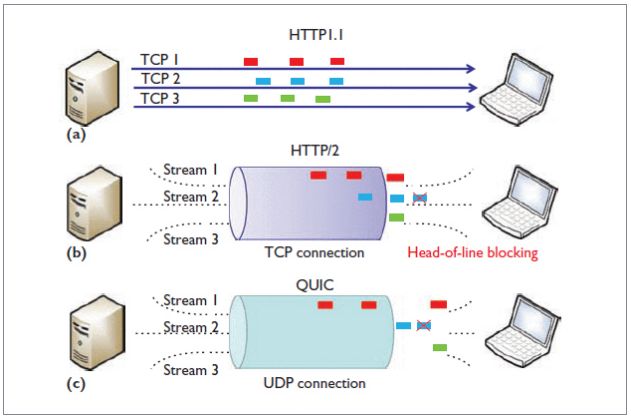

Figure 3: Multiplexing comparison [1].

Stream multiplexing is a method for sending multiple data streams over a single transport connection [1]. Figure 3 provides a comparison of multiplexing:

- **HTTP1.1** creates multiple TCP connections for different resources, as shown in Figure 3a.
- **HTTP/2** improves from HTTP1.1 and multiplexes streams over one TCP connection. However, the data are transmitted in order with TCP, and when one packet is lost on one stream, the packets on other streams are blocked. It leads to head-of-line blocking, as shown in Figure 3b.
- **QUIC** allows other streams to continue to receive packets instead of blocking all streams while waiting for packet recovery [18] because QUIC does not require ordered delivery of all packets, as shown in Figure 3c. Therefore, QUIC eliminates head-of-line blocking.

3.2. Connection Migration

The NAT routers track the TCP connections with the 4-tuple (source IP, source port, destination IP, and destination port). The NAT routers will rebind and change a source IP or port to the client according to the timeout. However, the NAT rebinding is more aggressive for UDP than TCP [3], [19]. Therefore, QUIC introduces a 64-bit Connection ID as an identifier for each connection, which is carried by the QUIC packet, as shown in Figure 2. Endpoints can use Connection ID to track QUIC connections without checking the 4-tuple [20]. At the beginning of the QUIC design, the Connection ID aims to avoid the problem of NAT rebinding. Currently, there is an extension of QUIC in progress, which supports generating routable QUIC Connection IDs [21].

3.3. Flow Control

QUIC provides two levels of flow control:

- **The stream-level flow control** limits the data sent on each stream. The initial limits of the streams are set during the handshake. A receiver can increase the limit periodically to accept more data on the stream by sending MAX_STREAM_DATA frames.
- **The connection-level flow control** defines the total bytes of stream data on all streams, and the limit of connection is determined according to the sum of bytes consumed on all streams [6].

3.4. Congestion Control

Similar to the TCP, QUIC chooses a pluggable congestion control interface, which at first is Cubic [22], to find the suitable algorithms. Notably, QUIC provides a different environment for congestion control than TCP [1]. To improve the design of TCP's sequence number, the QUIC packet is identified by the unique packet number [3]. The receiver reassembles the packet according to the stream offset. This separation of transmission order from delivery order avoids retransmission ambiguities. Moreover, QUIC provides more accurate network roundtrip time estimation with the delay information between when a packet is received and when the corresponding acknowledgment is sent [7]. Furthermore, QUIC supports up to 256 ACK blocks, making QUIC more resilient to reordering and loss than TCP [3]. Consequently, based on the same congestion control, QUIC detects and recovers from loss more efficiently [18].

3.5. 0-RTT Handshake

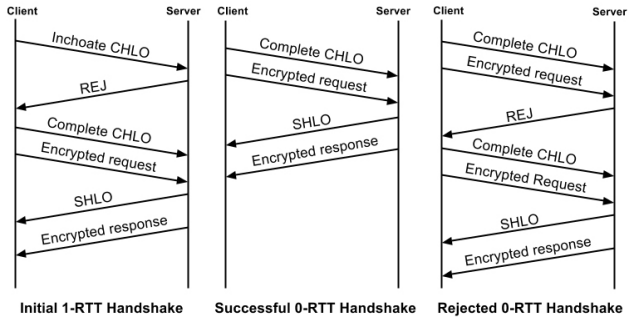

Figure 4: Timeline of QUIC's initial 1-RTT handshake, a subsequent successful 0-RTT handshake, and a failed 0-RTT handshake [3].

The Transport Layer Security (TLS) protocol provides a secure channel between two communicating peers. Compared to the previous version, TLS 1.3 adds a 0-RTT mode, saving a round trip at connection setup [23]. TCP that uses TLS 1.3 establishes a connection with at least 1-RTT. QUIC combines the cryptographic (TLS 1.3) and transport handshake to minimize the connection establishment time to 0-RTT. There are two types of handshake, as shown in Figure 4:

- **Initial 1-RTT handshake:** For the first connection to the server, the client sends the inchoate client hello (CHLO) message to the server. The server sends the reject message back, which contains valuable information for the subsequent connections [3].
- **Successful 0-RTT handshake:** For the client and server that had communicated before, the client can send complete CHLO with the encrypted request immediately [1].

4. QUIC RFCs

In this section, we discuss QUIC extensions, including RFC 9221 [13], RFC 9368 [8], and RFC 9369 [9]. Furthermore, we introduce RFC 9114 [10], which standardizes HTTP/3, a new version of HTTP that uses QUIC as the transport protocol.

4.1. RFC 9221

RFC 9221 [13] aims to extend QUIC to support the transmission of unreliable datagrams. The demand for transmitting unreliable data is increasing in some areas, such as audio/video streaming, gaming, and real-time network applications. QUIC is the better choice instead of UDP with the following advantages:

- QUIC provides a more precise loss recovery mechanism and more effective single congestion control.
- The application can use both a reliable stream and an unreliable flow within one connection, which benefits from the reduced handshake latency.
- QUIC provides a single congestion control for reliable and unreliable data.

Before RFC 9221 was published, Palmer et al. presented a simple extension to QUIC: ClipStream, used for video streaming [24]. This extension is designed based on a simple observation: not all frames in a video encoding scheme are equally important. ClipStream uses reliable transport for important frames (e.g. I-Frames) and unreliable transport for other frames (e.g. B- and P-Frames). Notably, ClipStream outperforms TCP and QUIC that not support the transmission of unreliable datagrams [24].

4.2. RFC 9368

Initially, QUIC does not provide a complete version negotiation mechanisms. The server can only reject the request from the client that uses an unacceptable version. RFC 9368 [8] updates this mechanism and defines the two types of version negotiation: compatible and incompatible version negotiation

Compatible version negotiation starts when the server knows how to parse the client's first packet, which contains the list of versions that the client knows its first packet is compatible with. The server must select one of these versions it supports as the negotiated version. Subsequently, the server converts the client's first packet into the negotiated version and replies to the client. Notably, version compatibility is not symmetric: A is compatible with B, but B may not be compatible with A.

Incompatible version negotiation happens if the server can not parse the client's first packet. The server sends a version negotiation packet to the client, which contains the server's offered versions. If the client does not find a version that it supports, the connection will be aborted. Incompatible version negotiation causes one more round trip than compatible version negotiation.

4.3. RFC 9369

RFC 9369 [9] defines QUIC version 2 to mitigate ossification concerns and exercise the version negotiation mechanisms, which are presented in Section 4.2. QUIC version 2 provides an example of the minimum changes necessary to specify a new QUIC version. The differences with QUIC version 1 are as follows:

- **Version Field** of long headers is 0x6b3343cf, generated by taking the first four bytes of the sha256sum of "QUICv2 version number".
- **Long header packet types** are different.
- **Cryptography changes** including the salt used to derive initial keys, HMAC-based key derivation function (HKDF) labels, and retry integrity tag.

For more details of the changes, we refer the readers to [9]. QUIC version 2 is not intended to replace version 1 and is compatible with version 1. Notably, a session ticket or token from a QUIC version 1 connection must not used to initiate a QUIC version 2 connection, and vice versa. Moreover, QUIC version 2 provides no different application functionalities than version 1. Furthermore, QUIC version 2 has no changes to the security.

4.4. RFC 9114

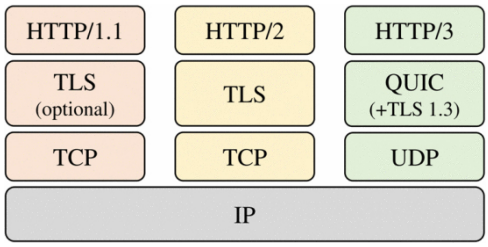

Figure 5: Protocol stack for different HTTP versions [25].

The Hypertext Transfer Protocol (HTTP), born in the early 90s, takes a dominant position in web protocols. Unlike the previous versions, HTTP/3 uses QUIC instead of TCP as the transport protocol, as shown in Figure 5. RFC 9114 [10] defines a mapping of HTTP semantics over QUIC. In this section, we describe how QUIC's features are mapped to HTTP/3 including the discovery of an HTTP/3 endpoint, QUIC streams, and per-stream flow control.

Discovery of an HTTP/3 endpoint: The server can announce the client with the Alt-Svc HTTP response header or the HTTP/2 ALTSVC frame using the "h3" ALPN token, which represents the availability of HTTP/3. The server may serve HTTP/3 on any UDP port. After receiving the Alt-Svc record, the client may establish a QUIC connection using HTTP/3.

QUIC streams: The stream data containing HTTP frame are carried by QUIC stream frames (Section 3.1). QUIC provides bidirectional and unidirectional streams. **Bidirectional streams** are only initiated by the client. Each bidirectional stream handles a pair of HTTP requests and responses. **Unidirectional streams** are used for a range of purposes. Two unidirectional stream types are defined:

- **Control streams** aim to manage other streams. At the beginning of the connection, each side of the endpoint must initiate a single control stream. A pair of single bidirectional streams ensures the performance and stability of data because it allows endpoints to send data as soon as possible.
- **Push stream** is designed for the server push feature introduced in HTTP/2. Server push allows the server to send resources that the client may need to the client without the client's request, as shown by Stenberg in [26].

Flow control: QUIC provides flexible flow control. The two levels of flow control, which are mentioned in Section 3.3, allow the developer to implement several mechanisms. The developer can send the MAX_STREAM_DATA frames in a linear, non-linear, or dynamic manner based on RTT estimates and application data consumption rate, as shown by Marx et al. in [27]. The mechanism of the QUIC flow control is still an open problem and can be improved.

5. Conclusion and Future Work

In this paper, we provide an overview of QUIC and its related RFCs. QUIC solves the head-of-line blocking. Furthermore, the connection can survive during the change of IP and port. Moreover, with the integration of TLS 1.3, the security is improved, and 0-RTT is facilitated to gain a faster handshake. Notably, QUIC shows excellent potential in extensibility. For unreliable connections, QUIC provides more choices for some areas, such as gaming and video streaming. With the promotion of the compatible version negotiation, version 2 of the QUIC is standardized, which provides an example of the new QUIC version. In Addition, QUIC has desirable features in transport for HTTP [10].

The IETF QUIC Working Group continues to contribute to the QUIC, and there are five directions:

- **The qlog**, an extensible high-level schema for a standardized logging format, is now used for the QUIC, HTTP/3, and QPACK events.
- **Connection migration** is going to be improved. The routable QUIC connection IDs are designed to route the packets with migrated addresses correctly.
- **Multipath extension** aims to enhance the usage of multiple paths for a single connection.
- **Control of delaying of acknowledgments** aims to improve connection and endpoint performance.
- **Reliable QUIC stream resets** allow sending the stream data up to a certain byte offset after resetting a stream.

We refer readers for more details to the in-progress documents [5], authored by the IETF QUIC Working Group.

References

[1] Y. Cui, T. Li, C. Liu, X. Wang, and M. Kühlewind, "Innovating transport with quic: Design approaches and research challenges," *IEEE Internet Computing*, vol. 21, no. 2, pp. 72–76, 2017.

[2] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better for what and for whom?" in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[3] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 183–196.

[4] [Online]. Available: https://w3techs.com/technologies/details/ce-quic

[5] [Online]. Available: https://quicwg.org/

[6] J. Iyengar and M. Thomson, "RFC 9000 QUIC: A UDP-based multiplexed and secure transport," 2021.

[7] J. Iyengar and I. Swett, "RFC 9002: QUIC Loss Detection and Congestion Control," 2021.

[8] D. Schinazi and E. Rescorla, "RFC 9368: Compatible Version Negotiation for QUIC," 2023.

[9] M. Duke, "RFC 9369: QUIC Version 2," 2023.

[10] M. Bishop, "RFC 9114: HTTP/3," 2022.

[11] M. Thomson, "RFC 8999: Version-Independent Properties of QUIC," 2021.

[12] M. Thomson and S. Turner, "RFC 9001: Using TLS to secure QUIC," 2021.

[13] T. Pauly, E. Kinnear, and D. Schinazi, "RFC 9221: An Unreliable Datagram Extension to QUIC," 2022.

[14] M. Thomson, "RFC 9287: Greasing the QUIC Bit," 2022.

[15] M. Kühlewind and B. Trammell, "RFC 9308 Applicability of the QUIC Transport Protocol," 2022.

[16] ——, "RFC 9312 Manageability of the QUIC Transport Protocol," 2022.

[17] C. Krasic and M. Bishop, "RFC 9204: QPACK: Field Compression for HTTP/3," 2022.

[18] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols," pp. 290–303, 2017.

[19] S. Hätönen, A. Nyrhinen, L. Eggert, S. Strowes, P. Sarolahti, and M. Kojo, "An experimental study of home gateway characteristics," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 260–266.

[20] [Online]. Available: https://blog.cloudflare.com/the-road-to-quic/

[21] [Online]. Available: https://quicwg.org/load-balancers/draft-ietf-quic-load-balancers.html

[22] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "CUBIC for fast long-distance networks," 2018.

[23] E. Rescorla, "Rfc 8446: The transport layer security (tls) protocol version 1.3," 2018.

[24] M. Palmer, T. Krüger, B. Chandrasekaran, and A. Feldmann, "The quic fix for optimal video streaming," pp. 43–49, 2018.

[25] M. Trevisan, D. Giordano, I. Drago, and A. S. Khatouni, "Measuring HTTP/3: Adoption and performance," pp. 1–8, 2021.

[26] D. Stenberg, "HTTP2 explained," pp. 120–128, 2014.

[27] R. Marx, J. Herbots, W. Lamotte, and P. Quax, "Same standards, different decisions: A study of QUIC and HTTP/3 implementation diversity," pp. 14–20, 2020.