QUIC Protocol: Solving TCP's Limitations

David Kim, d.h.kim@wustl.edu
(A paper written under the guidance of Prof. Raj Jain)

Download 

Abstract:

This paper introduces Quick UDP Internet Connections (QUIC), a protocol developed by Google to address the limitations of Transmission Control Protocol (TCP). This paper examines QUIC's architectural design, including the packet structure, multiplexing, and connection management using Connection IDs. Additional features including connection migration, loss detection, and congestion control are also analyzed in detail.

Keyword

QUIC, TCP, Protocol, Networking, UDP, Multiplexing, Connection Migration, Head of line Blocking, Loss Detection, TLS 1.3, Transport Layer, HTTP/3

Table of Contents:

- 1.0 Introduction
- 2.0 TCP vs QUIC
- 3.0 QUIC Protocol Overview
  - 3.1 QUIC Packet
  - 3.2 Multiplexing
  - 3.3 Packet Type
  - 3.4 Connection ID
  - 3.5 Security and Encryption
- 4.0 QUIC Reliability Features
  - 4.1 Connection Migration
  - 4.2 Loss Detection
  - 4.3 Congestion Control
- 5.0 Conclusion
- References
- Acronym

1.0 Introduction

QUIC was introduced in 2012 by Google when they ran into many performance-based issues on their web pages. After diagnosing the problem, they figured that the TCP has many significant limitations, especially with head of line blocking and the necessary three-way handshake required before transmitting data. To fix the problem, Google experimented and developed a new protocol

named QUIC which reduces the time it takes to establish a connection by using a much more efficient handshake method. QUIC is currently used as a transport layer for HTTP/3 which is currently the latest version of the Hypertext Transfer Protocol (HTTP) published by the Internet Engineering Task Force (IETF) back in 2022. Today, more applications and websites are adopting QUIC due to its performance benefits, with well known companies like Google, Cloudflare, YouTube, and Facebook deploying HTTP/3 on their websites to improve performance. In fact, W3Techs found that around 36.2% of all websites are currently using HTTP/3, showing rapid adoption of QUIC protocol in real-world applications [W3Techs25].

2.0 TCP vs QUIC

For the traditional TCP to establish a connection between the client and the server, a three-way handshake is required. The client sends a Synchronize (SYN) packet, in which the server sends an Synchronize-Acknowledge (ACK) packet, and the client replies with an ACK packet. On top of this, TCP also requires a separate Transport Layer Security (TLS) handshake for security, which adds additional round trips before any data can be sent. This will take 2 or more Round Trip Times (RTT) before the first data is sent, causing a substantial amount of delay [Chan22]. Instead, QUIC addresses this latency problem by combining the transport handshake with the encryption handshake into a single process, significantly reducing connection time. The details will be mentioned in Section 3.0. Another limitation with TCP is head of line blocking. Since TCP delivers data as a single stream, when a packet gets lost or delayed, all the packets behind must wait until the packet is recovered and received. This can increase reordering delays and consume a substantial number of resources, adding significant latency [Kosek21]. QUIC solves this problem through multiplexing.

3.0 QUIC Protocol Overview

Unlike TCP, QUIC has a different architectural design. As shown in the image below, TCP and TLS operate as a separate layer, thus requiring an additional handshake. With QUIC, TLS is integrated within the QUIC protocol itself, eliminating the need for an additional handshake. QUIC uses UDP as a transport layer, and each UDP datagram can send one or multiple QUIC packets.



Figure 1: Architectural Comparison of TCP and QUIC [Gaitonde23]

3.1 QUIC Packet

A QUIC packet contains two main components, the header and the payload. The invariant properties must always be present in the first packet of the datagram [RFC 8999], while the rest of the packets can have different formats based on their version. There are two types of headers, Long Header and Short Header. Long Headers are specifically designed for packets before the connection has been established, packets including Initial, 0-RTT, Handshake, Retry, and Version Negotiation packets. Short Headers are used after the connection has been fully established for ongoing 1-RTT data transmission. In QUIC Version 1 from Request for Comments (RFC) 9000, the format for Long and Short Header goes like this:

Table 1: RFC 9000 Long Header

Field Name	Bit Length	Description
Header Form	1	Set as 1 to indicate Long Header packet.
Fixed Bit	1	Set as 1 for all QUIC v1 packets. 0 may represent Version Negotiation Packet and is invalid for v1.
Type-Specific Bits	4	Reserved bits that depends on packet type.
Version	32	QUIC version number.
Destination Connection ID Length	8	Length of destination ID.
Destination Connection ID	0-160	ID for the receiver.
Source Connection ID Length	8	Length of source ID.
Source Connection ID	0-160	ID for the sender.

Table 2: RFC 9000 Short Header

Field Name	Bit Length	Description
Header Form	1	Set as 0 to indicate Short Header packet.
Fixed Bit	1	Set as 1 for all QUIC v1 packets. 0 may represent Version Negotiation Packet and is invalid for v1.

Spin Bit	1	Enables latency monitoring, measuring time of RTT connection.
Reserved Bits	2	Reserved bits where it should be set as 0.
Key Phase	1	Identify the protection keys it is used for the packet.
Packet Number Length	2	Length of packet number.
Destination Connection ID	0-160	ID for the receiver.
Packet Number	8-32	Packet Number

After the header, the payload follows along, and it contains the actual data the sender is trying to transmit to the receiver. QUIC organizes its payload into different frames where each frame serves a different purpose in the protocol. Each packet's payload can contain multiple frame types, including PADDING frames, ACK frames, STREAM frames, CRYPTO frames, and others.

All frames follow a general structure where it begins with the type field, and depending on the frame type, it follows by its own specific fields. One thing to note is that QUIC packet uses a variable-length encoding for integer variables, meaning that the bit length is not set but rather adjusted based on how much value it needs to be represented, making it much more efficient when encoding. The most common frame is a stream frame which is considered the frame where application data is sent [RFC 9000]. The format goes as follows:

Table 3: Stream Format

Field Name	**Description**
Type	Identifies which type of stream it is using (0x8 to 0xf for regular stream)
Stream ID	A unique ID for the stream
Offset	Byte offset in the stream
Length	Length of the stream
Stream Data	The actual data being sent

Having a frame enables QUIC to handle multiple different data streams simultaneously, and the process will be discussed in the next section.

3.2 Multiplexing

Mentioned in the previous section, QUIC solves TCP's head of line blocking through the usage of multiplexing. Multiplexing organizes the payload into different streams shown in the figure below, where each stream is identified by the Stream ID [RFC 9000]. When a packet belonging to one stream gets lost, only the stream with that ID has to wait for recovery while the other streams continue to deliver the data to the receiver without interruption. By isolating packet loss to its specific stream, it will significantly reduce the latency, allowing QUIC to perform better than its competitors.



Figure 2: Illustration of Multiplexing [Gaitonde23]

3.3 Packet Type

As discussed earlier, QUIC v1 has 5 different packet types, each containing a specific purpose when establishing a connection.

Version Negotiation Packet is created by the server when it realizes that the client's QUIC version does not match with its own. It informs the client of the version it is required to successfully send packets to the server [RFC 9000].

Initial Packet, as the name suggests, is the first packet that is sent to the server to establish a connection. This packet transports CRYPTO frames which contain the key exchange information, and the server acknowledges with its own initial packet. This packet stops once both sides start using Handshake packets [RFC 9000].

Handshake Packet sends encrypted handshake messages and acknowledgments between the server and the client.

0-RTT contains early data that is sent to the server before the handshake is completed. This packet can only be used when the client has previously connected to the server, as it requires the keys for encryption [TLS].

Retry packet is sent when a server wants to validate the client's address before establishing a connection. The server creates an address validation token and sends it as a retry packet. When receiving the retry packet, the client has to start over and send a new initial packet that has this token [RFC 9000].

3.4 Connection ID

Connection ID is part of the QUIC header that is used to identify connections. QUIC can allow client and server to be connected even when there are changes at the underlying layer such as User Datagram Protocol (UDP) and Internet Protocol (IP) because QUIC uses Connection ID for a connection instead of the other layers. Each endpoint selects the Connection ID that its peer should use when sending packets where each ID should be unique to prevent packets being misrouted to the wrong endpoints.

Connection ID has two types, Source Connection ID and Destination Connection ID. Long packets as we discussed earlier requires both ID to establish a connection, while the short packets typically only require Destination Connection ID as the connection has already been previously established. A zero length connection ID is also possible, and it is specifically only used when the connection ID is not needed to route the packets to the right endpoints [RFC 9000].

Beyond identifying connections, QUIC is known for having security as a core feature, which is explored in the following section.

3.5 Security and Encryption

As seen in Figure 1, the security feature for TLS is a separate layer than TCP. This means that TCP connection must be fully established before TLS encryption can begin. However, with QUIC, TLS 1.3 is built directly into the protocol, meaning that the encryption happens simultaneously with the connection establishment. This integration will reduce time by 1 to 2 RTT compared to TCP, depending on the TLS version TCP uses (2-RTT savings for 1.2 and 1-RTT savings for 1.3). QUIC does this by sending the handshake messages with the CRYPTO frame when establishing a connection. The client first sends a Client Hello message that contains key share information used for exchanging cryptographic keys. The server then responds with Server Hello message that has its own key share information, allowing both sides to exchange encryption keys in 1 Round Trip [TLS].

As mentioned before, TLS 1.3 also allows 0-RTT in which encrypted data can be sent in the very first packet only when a client and the server have previously made a connection. Although this eliminates handshake latency, 0-RTT comes with a security tradeoff. 0-RTT is vulnerable to replay attacks where an attacker can intercept and retransmit the same 0-RTT data to the server, and the server cannot verify whether if it is from the client [TLS]. Therefore, 0-RTT is usually not recommended if security is the top priority.

Other than a faster connection establishment, QUIC also provides a stronger privacy through heavy encryption. In TCP, only the payload is encrypted while the headers remain visible in plaintext. QUIC encrypts nearly all packet contents, including frame types [Dellaverson21]. Only the minimal information required for routing remains unencrypted. Therefore, it offers a better privacy protection by hiding the metadata from potential attackers. Furthermore, QUIC's encryption limits protocol ossification. Ossification occurs when middleboxes like firewalls inspect and adapt based on the structure of the protocol. TCP has been ossified because middleboxes constantly read and modify the headers, making it harder to change or update without breaking its existing structure.

QUIC prevents middleboxes from inspecting, thus it will have lower dependence on its internal mechanisms. This will make it much easier to add and update new features without much modification [Dellaverson21].

With these security features in place, QUIC provides additional reliability features that enhance its performance over TCP.

4.0 QUIC Reliability Features

QUIC contains many features that provide reliable and efficient data transmission. These features address the limitations of TCP while showcasing how QUIC can be useful for modern applications.

4.1 Connection Migration

As mentioned earlier, Connection ID enables consistent connection even when there are changes at the underlying layers such as IP and Ports. This concept is known as Connection Migration, and it allows QUIC to be connected without requiring another establishment.

This mostly happens when the client switches their network to another. The most common example is when a mobile device switches from WiFi to cellular data [Connection]. This causes the IP address and port to change. This will affect TCP connection because it uses the source IP and port and destination IP and port. This means that TCP connection is terminated, and a new connection must be reestablished until the IP and ports are updated so that the server and client can be successfully reconnected again. With QUIC, it uses connection ID for identifying connections rather than relying on IP and ports. The connection ID remains consistent even when there are changes to the IP and port. All the server needs to do is validate the incoming packets from a new network path, and transition over to the new path once the validation is confirmed.

4.2 Loss Detection

When packets have been lost while transmitting to the endpoints, re-transmission is required for the packets so that either the server or client receives the complete message. QUIC detects these lost packets by both acknowledge-based and timer-based methods. The receiver sends an ACK frame to the sender once it obtains a packet from them. If the sender misses an ACK frame from a particular packet, it is considered lost, and retransmission is required [RFC 9002].

QUIC also uses a Probe Timeout (PTO), which is often calculated based on the expected timeframe of RTT. When ACK frame is not received within the timeframe, the PTO triggers into sending one or two probe packets to elicit acknowledgment from the receiver. If the acknowledgment is still not received, PTO deems it as being lost, and it recovers the lost packets or acknowledgment. It does this by retransmitting the missing frames in a new packet with a new sequence number. Furthermore, a lost packet affects only that specific stream and not the entire stream connection, thus preventing the head of line blocking [Haynes23].

4.3 Congestion Control

Congestion Control is a way to regulate network traffic so that there is no overloading in the network path, which can cause severe packet loss and delays. QUIC implements congestion control per path, meaning that the congestion controller remains the same even when packets are sent to other paths [RFC 9002].

The algorithms in which the congestion controller uses can vary, but the most typical algorithm it uses is similar to TCP NewReno and Cubic which are control algorithms that are used in TCP. However, different algorithms can be used depending on the conditions and application requirements the network wants to have. With the improvements in the loss detection, QUIC can use the packet number to determine between network congestion and other sources of delays, thus giving a more accurate congestion control [RFC 9002].

5.0 Conclusion

The table below summarizes the main differences between TCP and QUIC discussed throughout the paper:

Table 4: TCP vs QUIC Overall Comparison

Feature	**TCP + TLS**	**QUIC**
Connection Establishment	2-3 RTT	1 RTT (0-RTT on resumption)
Transport Protocol	TCP	UDP with QUIC
Encryption	Separate Layer of TLS	Built-in TLS 1.3
Multiplexing	Single stream (Head of line blocking)	Multiple independent streams
Connection Migration	Not supported	Supported (through Connection ID)
Header Encryption	Not Encrypted (Plaintext)	Encrypted
Loss Detection	ACK-based + Retransmission Timeout	ACK-based + Probe Timeout

QUIC addresses the inherent limitations of TCP that has affected many deployments for years. With QUIC's new features such as TLS 1.3, connection migration, multiplexing, and better congestion control, QUIC provides a significant improvement in performance, especially for modern data-intensive applications. These benefits have led to QUIC's adoption as the transport layer for HTTP/3, and later was standardized by IETF in 2022 [RFC 9114]. Research demonstrates that QUIC reduces latency for Google Search by 3.6-8.0% for users and decreases rebuffering rates on YouTube by 15-18%, showcasing its real-world benefits for modern web services

[Langley17]. However, QUIC also comes with certain limitations. Many firewalls and security appliances block QUIC traffic, forcing it to fall back to TCP [Andrea]. In addition, QUIC runs within the user space, resulting in a much higher CPU cost than TCP [Oku20].

References

[RFC 8999] M. Thomson, "RFC 8999 Version-Independent Properties of QUIC", Internet Engineering Task Force, 2021, https://www.rfc-editor.org/rfc/rfc8999.html

[RFC 9000] J. Iyengar and M. Thomson, "RFC 9000 QUIC: A UDP-Based Multiplexed and Secure Transport", Internet Engineering Task Force, 2021, https://www.rfc-editor.org/rfc/rfc9000.html#name-loss-detection-and-congesti

[RFC 9002] J. Iyengar and I. Swett, "QUIC Loss Detection and Congestion Control", Internet Engineering Task Force, 2021, https://datatracker.ietf.org/doc/html/rfc9002

[RFC 9114] M.Bishop and Ed.Akamai, "HTTP/3", Internet Engineering Task Force, 2022, https://datatracker.ietf.org/doc/html/rfc9114

[Langley17] Adam Langley et al., "The QUIC Transport Protocol: Design and Internet-Scale Deployment", ResearchGate, 2017, https://www.researchgate.net/publication/318914953_The_QUIC_Transport_Protocol_Design_and_Internet-Scale_Deployment

[Gaitonde23] Animesh Gaitonde, "How QUIC Is Displacing TCP for Speed and Security?", Engineering At Scale, 2023, https://engineeringatscale.substack.com/p/how-quic-is-displacing-tcp-for-speed

[Oku20] Kazuho Oku and Jane Iyengar, "QUIC matches TCP's efficiency, says our research", Fastly, 2020, https://www.fastly.com/blog/measuring-quic-vs-tcp-computational-efficiency

[Haynes23] Robert Haynes, "A Primer on QUIC Networking and Encryption in NGINX", F5, 2023, https://www.f5.com/company/blog/nginx/primer-quic-networking-encryption-in-nginx

[Kosek21] M. Kosek, T. Shreedhar, and V. Bajpai. "Beyond QUIC v1: A First Look at Recent Transport Layer IETF Standardization Efforts", IEEE Communications Magazine, vol. 59, no. 4, Apr. 2021, pp. 24-29. Crossref, https://doi.org/10.1109/mcom.001.2000877

[Dellaverson21] J. Dellaverson et al., "A Quick Look at QUIC", 2021, https://web.cs.ucla.edu/~lixia/papers/UnderstandQUIC.pdf

[Chan22] Alysa Chan, "TCP and TLS handshake: What happens from typing in a URL to displaying a website? (Part 2)", Medium, 2022, https://medium.com/@alysachan830/tcp-

and-tls-handshake-what-happens-from-typing-in-a-url-to-displaying-a-website-part-2-243862438cd9

[W3Techs25] "Usage statistics of HTTP/3 for websites", W3Techs, 2025, https://w3techs.com/technologies/details/ce-http3

[Andrea] Harris Andrea, "What is QUIC - This Modern Internet Protocol Makes Firewalls Blind", Networks Training, n.d., https://www.networkstraining.com/what-is-quic-protocol/ [Discussion on firewall blockage of QUIC protocol]

[Connection] "Connection Migration", n.d., https://quic-go.net/docs/quic/connection-migration/ [Documentation of connection migration functionality in QUIC implementation]

[TLS] "TLS 1.2 vs. 1.3—Handshake, Performance, and Other Improvements", n.d., https://www.catchpoint.com/http2-vs-http3/tls1-2-vs-1-3 [Comparison between TLS 1.2 and TLS 1.3 handshake]

Acronyms

QUIC - Quick UDP Internet Connections
TCP - Transmission Control Protocol
UDP - User Datagram Protocol
TLS - Transport Layer Security
RTT - Round Trip Time
ACK - Acknowledgment
SYN - Synchronize
CRYPTO - Cryptographic
PTO - Probe Timeout
V1 - Version 1
RFC - Requests for Comments
HTTP - Hypertext Transfer Protocol
IETF - Internet Engineering Task Force

Last modified on November 17, 2025
This and other papers on recent advances in networking are available online at
http://www.cse.wustl.edu/~jain/cse5700-25/index.html
Back to Raj Jain's Home Page