Advantages and Disadvantages of QUIC/UDP protocol

QUIC was introduced by Google, it is a protocol to overcome the limitation of TCP protocol which was widely used in HTTP. QUIC has rapidly increased in the number of QUIC-enabled IP address from around 2 million addresses in 2021 to 8 million addresses in 2023 (Nguyen et al, 2025, Introduction, para. 2), Cisco Visual Networking Index predicts a 23% Compound Annual Growth Rate in global IP traffic from 2021 to 2026, and QUIC is expected to become a dominant web content delivery protocol in the future (Tauqeer et al, 2024, Introduction, para. 4). QUIC protocol has been adapted and widely used by many tech giants, over 30% of Google's Server network traffic is QUIC packets, it is also developed on Chrome, both desktop and mobile, QUIC already took up to 7% of internet traffic (Arisu et al, 2020, Introduction, para. 1). QUIC is a protocol developed by Google, it was formally adopted as a standard by IETF in May 2021; however, the engineers at Google started developing it in 2012. Originally, it was just an experiment to increase performance of Google web server (Petryschuk, 2024, What is QUIC protocol, paras. 1-2). Unlike TLS, QUIC protocol was built on UDP protocol. QUIC packet can be categorized into 2 types, they are long header packet and short header packet. Long header packet is mainly used for establishing connection, while short header packet is used for transmitting actual data to optimize the efficiency (Tanveer, 2024, The QUIC packet structure, paras. 1-3; f5.com, n.d., Overview of a QUIC network).

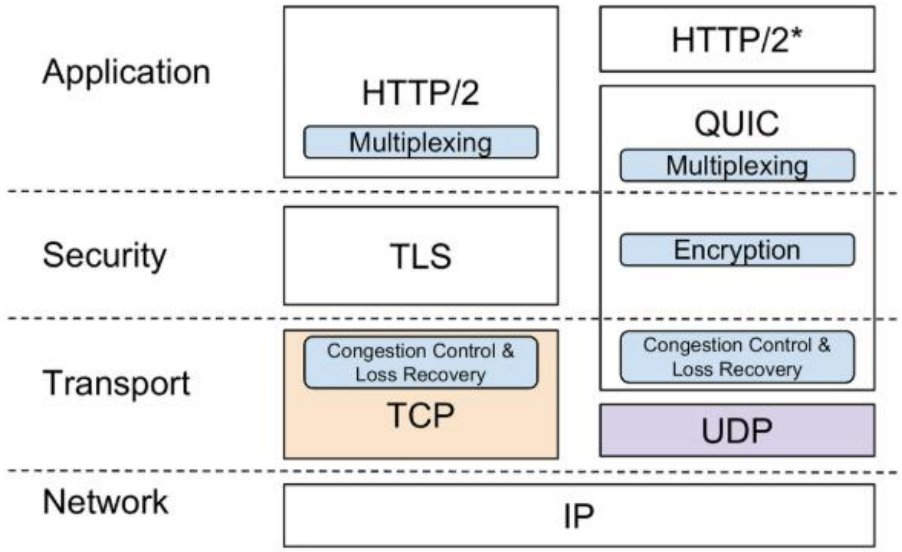

Figure 1: The structure of TLS and QUIC. From Arisu, 2020

Because QUIC is built on UDP, it offers a lower latency than TLS, when establishing a new connection, TCP itself requires 1.5 round-trip time (RTT) for the handshake, and with TLS, it could take up to 2-3 RTT for the 3-way handshake and to transmit security certificates for encrypting and decrypting before the data is actually send (Arisu et al, 2020, Introduction, para. 3). In case if the client connects to server in the first time ever, QUIC only needs 1 RTT for Client Hello and security certificates from the server, because all essential information to establish a connection between server and client and encryption information are sent all in once from client to server without having to wait a response from server like in TLS. In case the client already connected to the server before, it caches the server's information and sends the server's information with the request to the server, then the server can just send the response with the requested data back to the client, it is 0 RTT connection, when there is no handshake needed between client and server if they already did that before (Zhang et al, 2021, The QUIC protocol, para. 2; Perna, 2022, Background and related work, para. 3). QUIC is also reliable for HTTP/3 because it combines the characteristics of UDP and TCP together, while UDP is a lightweight protocol, it does not require handshake before transmitting

data, but it is unreliable because it can not prevent packet loss, while TCP can handle packet loss, but it require a high latency, complicated handshake, especially when using TLS; as well as, when a packet is lost, TCP will block all other packet to reach client before the lost packet is retransmitted to the client. QUIC is built on UDP but it can still handle packet loss without blocking the all other packets, to do this, QUIC uses multiplexed streams to transmit packets, multiplexing in QUIC allow the packet to be transmitted on multiple ports, while in TCP, only port 443 is used to transmit packets, each stream is independent to others, when a packet is lost, QUIC will just hold the packet to that particular stream and retransmit the packet in the stream so other steams can just deliver packets without being affected (Arisu et al, 2020, Introduction, para. 2; IONOS editorial team, 2023, What advantages does QUIC offer, paras. 2-3). Moreover, QUIC implements a flow control for the streams and connection to prevent buffer overflows. The maximum size of data allowed during transmission is sent to the sender in the 1-RTT handshake, MAX_STREAM_DATA and MAX_DATA frame is sent from the receiver, guiding the sender about the maximum data it can send in a stream and in a connection. If the maximum data in a stream or in a connection is reached, the packet will be blocked until new MAX_STREAM_DATA or MAX_DATA frame is advertised (Lee & An, 2022, Related work, para. 2).

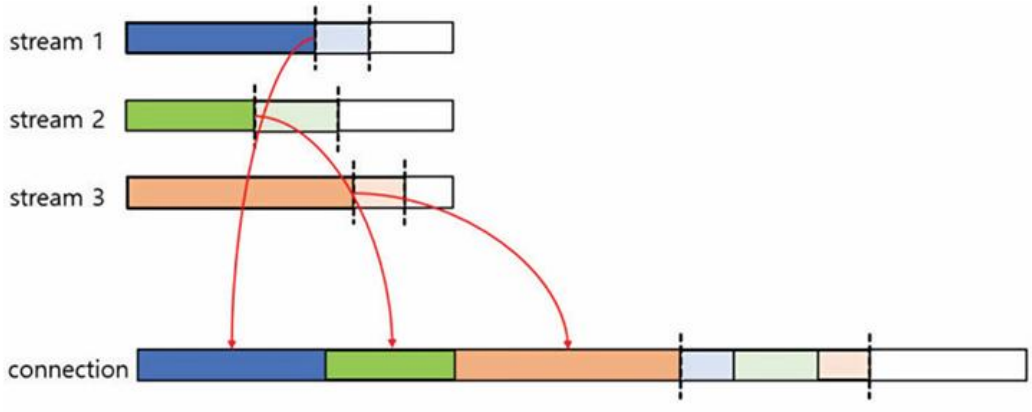

Figure 2: Visualization of maximum allowed data in streams and in connection. From Lee & An, 2022, CC BY 4.0

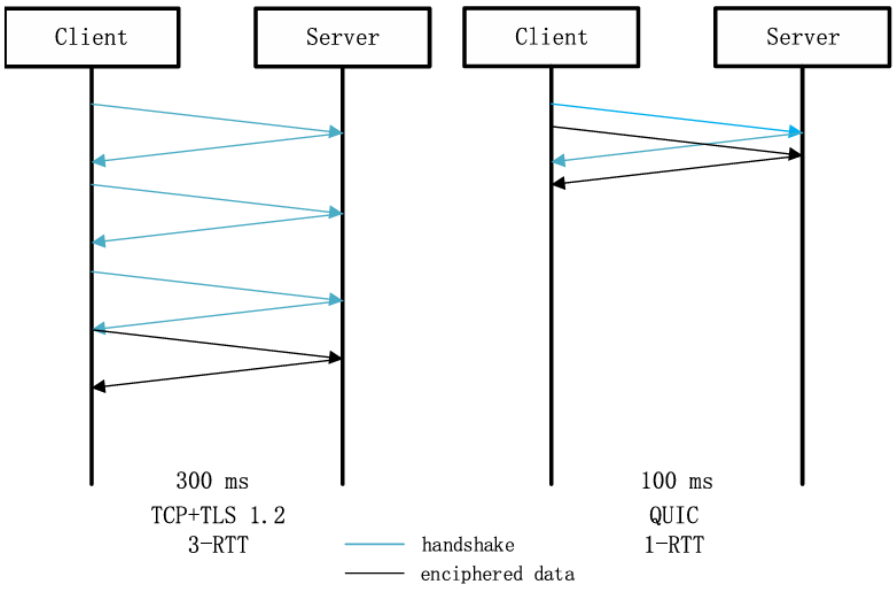

Figure 3: Handshake process of TLS and QUIC. From Zhang et al, 2021

QUIC still has some disadvantages, the main problem of it is overhead. It is like a trade off while TLS/TCP offers low overhead but longer delay (longer RTT), while QUIC offers lower latency but more likely to be overhead. TLS/TCP control the packets such as ACKs happens within the OS kernel, QUIC uses packets control in the userspace; as a result, system call is required to send and receive packets, these system calls could have operations such as context switching, buffer allocation and copying, and socket lockup (Kumar et al, 2025, Introduction, para. 5). Moreover, QUIC also has vulnerability related to 0-RTT connection, when the server allows client to get data from server without having to perform handshake if the client already cached server's information and send it to the server inside the packet. Even though the comparison between TLS/TCP and QUIC/UDP on HTTP/2 shows that TLS/TCP has a greater number of vulnerabilities which could cause more negative impact on the server than QUIC/UDP; however, QUIC/UDP is not perfectly secured, it can still be attacked by replay attack and packet manipulation attack, especially in 0-RTT connection (Murthy et al, 2022, Background and related work, paras. 2-3). In 0-RTT connection, the attacker can use man-in-middle method to steal the packet sent from client to server with the

confidential information to allow the client to establish 0-RTT connection with the server, the attacker can use the stolen packet to impersonate the client and then to exploit the server. This type of attack is difficult to resolve because if the server require client to perform handshake before establishing connection, it will be against of QUIC's purpose is to reduce the latency caused by traditional TCP 3-way handshake. The second type of attacking is the attacker sends minimum amount of data just enough to keep the connection open; this can exhaust the resource on the server and hard to be detected because unlike DoS/DDoS attack, the traffic generated by the attack is very low. Another type of attack is the DoS-style attack, when the attacker opens a large number of streams and the attacker does not send all the necessary data to the server to keep the stream open, this makes the server to track and maintain a very large number of streams (Chatzoglou et al, 2023, Preliminaries, paras. 1-3; RocketMe Up Cybersecurity, 2024, Security Challenge with HTTP/3 and QUIC). Although QUIC offers a significant lower latency than TLS; however, some latency-sensitive applications still can not afford the delay and drop the lost packet. In TCP, a packet is marked lost after few RTTs; in QUIC protocol, it does not wait for few RTTs, a packet will be marked as lost if it meets 2 criteria: if the packet has been sent for at least $\frac{9}{8}$ * RTT and an ACK is received for at least 1 packet after the lost packet was sent (Michel & Bonaventure, 2024, QUIC and loss recovery techniques, para. 2).

QUIC is a represent for the development of the network industry, it offers numerous benefits, such as low latency, multiplexed streams, quick and simple 1-RTT or 0-RTT handshake process. However, it comes with trade-offs such as overhead, and new type of attack can be performed on network traffic with QUIC protocol, and these attacks may require more effort to detect and resolve. Overall, using QUIC in network traffic between server and client can reduce the delay time, increase security; however, it will require more time, cost, and effort to administrate the network traffic.

References

Arisu, S., Yildiz, E., & Begen, A. C. (2020). *Game of protocols: Is QUIC ready for prime time streaming? International Journal of Network Management, 30*(3), Article e2063. https://doi.org/10.1002/nem.2063

Chatzoglou, E., Kouliaridis, V., Karopoulos, G., & Kambourakis, G. (2023). *Revisiting QUIC attacks: A comprehensive review on QUIC security and a hands-on study. International Journal of Information Security, 22*, 347–365. https://doi.org/10.1007/s10207-022-00630-6

CloudFlare. (n.d.). *What Happens in a TLS Handshake? | SSL Handshake | Cloudflare.* Cloudflare. https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/

f5.com. (n.d.). *What Are QUIC and HTTP/3?* F5, Inc. https://www.f5.com/glossary/quic-http3

IONOS editorial team. (2023, March 1). *QUIC: What is behind the experimental Google Protocol?* IONOS Digital Guide. https://www.ionos.ca/digitalguide/hosting/technical-matters/quic-the-internet-transport-protocol-based-on-udp/

Kumar, P., & Dezfouli, B. (2025). *kQUIC: A Kernel-Based Quick UDP Internet Connections (QUIC) Transport for IoT. IEEE Access, 13*, 100392–100404. https://doi.org/10.1109/ACCESS.2025.3578120

Lee, S., & An, D. (2022). *Enhanced Flow Control for Low Latency in QUIC. Energies (Basel), 15*(12), Article 4241. https://doi.org/10.3390/en15124241

Michel, F., & Bonaventure, O. (2024). *QUIRL: Flexible QUIC Loss Recovery for Low Latency Applications. IEEE/ACM Transactions on Networking, 32*(6), 5204–5215. https://doi.org/10.1109/TNET.2024.3453759

Murthy, A., Asghar, M. R., & Tu, W. (2022). *Towards a data-driven framework for optimizing security-efficiency tradeoff in QUIC. Security and Privacy, 5*(1). https://doi.org/10.1002/spy2.184

Nguyen, T. T., Vu, M. H., Le Nguyen, P., & Nguyen, K. (2025). *Analysis and visualization of QUIC protocol deployment in Vietnam. IEICE COMMUNICATIONS EXPRESS*, 1–4. https://doi.org/10.23919/comex.2025XBL0091

Perna, G., Trevisan, M., Giordano, D., & Drago, I. (2022). *A first look at HTTP/3 adoption and performance. Computer Communications, 187*, 115–124. https://doi.org/10.1016/j.comcom.2022.02.005

Petryschuk, S. (2024, October 23). *What is QUIC? Everything You Need to Know | Auvik Networks*. Auvik Networks Inc. https://www.auvik.com/franklyit/blog/what-is-quic-protocol/

RocketMe Up Cybersecurity. (2024, November 12). *The Security Challenges of HTTP/3 and QUIC — What You Need to Know*. Medium. https://medium.com/@RocketMeUpCybersecurity/the-security-challenges-of-http-3-and-quic-what-you-need-to-know-7c1996cbc502

Tanveer, M. (2024, July 24). *Understanding QUIC Protocol: A Detailed Look at Packet Structure*. Threatpointer. https://mdtanveer.com/understanding-quic-protocol-a-detailed-look-at-packet-structure

Tauqeer, M., Gohar, M., Koh, S. J., & Alquhayz, H. (2024). *Use of QUIC for Mobile-Oriented Future Internet (Q-MOFI). Electronics [Basel], 13*(2). http://dx.doi.org.conestoga.idm.oclc.org/10.3390/electronics13020431

Zhang, J., Yang, L., Gao, X., Tang, G., Zhang, J., & Wang, Q. (2021). *Formal analysis of QUIC handshake protocol using symbolic model checking. IEEE Access, 9*, 14836–14848. https://doi.org/10.1109/ACCESS.2021.3052578