SMB over QUIC: A Performance Evaluation

Ognjen Mitrović, Vedran Tadić
University of Applied Sciences , Zagreb, Croatia
omitrovic@tvz.hr, vtadic@tvz.hr

***Abstract*—The ever-growing usage and adaptation of HTTP/3 in web traffic leverages Quick UDP Internet Connections (QUIC) as a transport protocol. QUIC is widely recognized for reduced latency, enhanced security, and improved performance compared to TCP or UDP. Although known primarily for HTTP/3, QUIC also exhibits versatility for other applications, such as file transfers. In this paper, Microsoft's implementation of SMB over QUIC is evaluated and compared with traditional SMB over TCP. The measurements emphasize protocol behavior under low network congestion and examine resource utilization, including CPU and memory load, on both client and server. By contrasting SMB over QUIC with SMB over TCP, potential advantages and trade-offs of adopting QUIC for enterprise file sharing are identified. The results offer insights into the performance characteristics of QUIC within file transfer scenarios and contribute to the overall understanding of QUIC's practical applications, providing guidance for optimizing file transfer systems.**

Keywords— QUIC, SMB, Performance, Measurement, TCP, Evaluation, Windows Server, Microsoft, Protocol, Network

I. INTRODUCTION

The Server Message Block (SMB) protocol underpins file-sharing infrastructures in many Windows-based enterprise environments. Traditionally, SMB has operated over TCP, benefiting from longstanding kernel-level optimizations and hardware offloads on Windows. Meanwhile, Quick UDP Internet Connections (QUIC)—originally developed for HTTP/3—has proven valuable in accelerating web content delivery and reducing handshake overhead, particularly under higher latency or lossy conditions. [1][2] [3][4]

Previous studies have analyzed QUIC performance primarily for HTTP/3 or HTTP/2 scenarios and have compared it to TCP performance in terms of page load times, round-trip times, and packet-loss ratios [2][3][5][6][7]. However, none of these efforts have examined file transfer performance using SMB over QUIC. Microsoft introduced SMB over QUIC as built in protocol for latest Windows Server 2025 operating system which aims to build on QUIC's key benefits, such as firewall-friendly connectivity using UDP port 443 instead of commonly blocked TCP port 445 and integrated encryption, to enable remote file access – including cloud providers platforms - without requiring a dedicated VPN. [8] Windows QUIC implementation allows multiplexing protocol usage on UDP port 443 (like SMB and HTTP) using Application Layer Protocol Negotiation ALPN.

Although QUIC achieves notable success in HTTP/3, it remains uncertain whether similar benefits translate directly to a stateful protocol such as SMB. [3][5] Unlike HTTP, which often processes short, stateless transactions, SMB depends on multi-step negotiations, persistent state, and authentication procedures that may overshadow QUIC's streamlined handshake. In this paper, SMB over QUIC is compared with SMB over TCP in a low-latency Hyper-V environment. Throughput is measured for both ephemeral and persistent sessions, CPU usage is assessed on both client and server, and memory consumption is observed. Explanations are provided regarding how the Hyper-V host CPU can become a bottleneck for sustained large-file transfers and why SMB over QUIC behaves differently from the commonly referenced QUIC-over-HTTP(S) scenario.

II. MEASUREMENT METHODS

A. Test Environment

The testings were conducted on a single physical host running Windows 11, featuring an Intel i7-12700H CPU with 32 GB of RAM and 1 TB SSD. Two Generation 2 virtual machines were deployed on this host, each assigned a single virtual adapter connected to a private Hyper-V switch. Internet-based traffic was excluded by isolating the Hyper-V network, and only minimal background processes were run on both VMs.

The client machine (VM1) was configured with a default Windows 11 installation, allocated 4 vCPUs and 6 GB of RAM. In contrast, the server (VM2) ran a default Windows Server 2025 installation, with 4 vCPUs and 4 GB of RAM. The additional 2 GB of RAM on the client was selected to accommodate the overhead from script execution and because the Windows client operating system tends to require more memory than Windows Server due to additional built-in components. Advanced SMB client and server settings were left at their defaults. During early experimentation with QUIC SMB parameters, an attempt to increase the connection count per RSS network interface from the default 4 to 8 (using the 'Set-SMBClientConfiguration' cmdlet) occasionally caused PowerShell to hang during QUIC session removal. This was ultimately resolved by performing full client system restart. Unfortunately, Microsoft documentation currently provides limited guidance on many of these configuration parameters [8][9].

SMB Multichannel remained enabled by default in the operating system, despite the absence of multiple network adapters on each VM. In practice, it occasionally created

2025 MIPRO 48th ICT and Electronics Convention | 979-8-3315-3597-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/MIPRO65660.2025.11132048

multiple QUIC streams (up to four) over a single interface during large file transfers. This behavior was observed using QUIC Performance Diagnostic counters set inside Performance Monitor tool. Antivirus services were disabled on both VMs to avoid any overhead related to file scanning.

No Active Directory domain environment was used, causing NTLM to serve as the authentication protocol. This NTLM exchange was tunneled through QUIC's TLS 1.3 connection. A self-signed certificate was generated on the server for QUIC session establishment, while client certificate verification was disabled (using Requireclientauthentication parameter of New-SMBServerCerificateMapping cmdlet), since no corresponding client certificate had been created. Because no DNS infrastructure was present, host's files were configured on both VMs to map host names for certificate validation and name resolutions.

A limited set of packet captures was performed in Wireshark and decrypted with a custom Frida Python script [10]. However, these captures were not incorporated into the final measurements to avoid any overhead that could interfere with PerfMon data collection.

This setup was designed to reduce complexity and is believed to provide future researchers with a more straightforward path to setting up their test environments, as it minimizes the number of required components and system resources.

B. Test scenarios and file sizes

Four primary scenarios were tested, based on whether SMB used TCP or QUIC, and whether the session was re-established for each file transfer (ephemeral) or retained for multiple file transfers (persistent). In the ephemeral mode (with session overhead), each file copy involved creating a new SMB mapping, transferring the file, and subsequently removing the mapping. In the persistent mode (without session overhead), a single SMB mapping was created once, used for all transfers, and removed only after all tests had concluded.

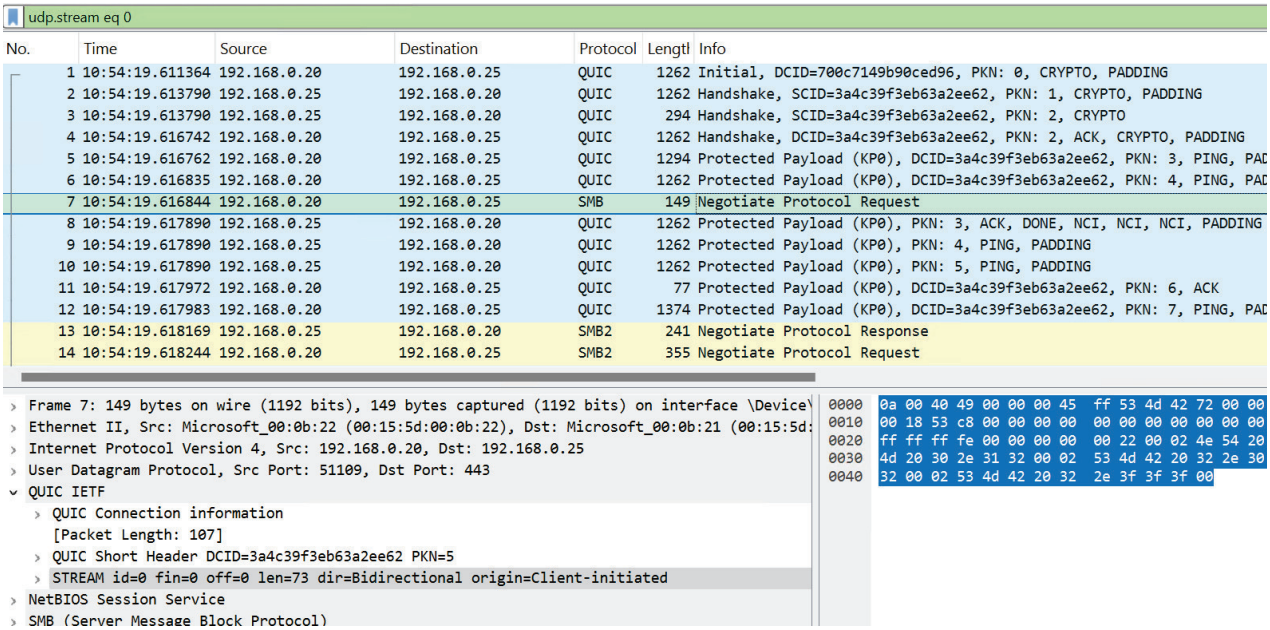

Figure 1. Decrypted SMB over QUIC packets captured in Wireshark

These two session models were designed to reflect distinct, realistic usage patterns. Within many enterprise environments, a user logs on in the morning and maintains an SMB session active for the entire workday, without requirement for additional negotiations. However, certain scripts, applications or microservices may connect briefly, copy one or two files, and then disconnect repeatedly, simulating short-lived connections. By creating and removing the SMB share for each file, ephemeral usage was reproduced. Although a single client VM was used for the iteration, the server processed each as a distinct session over time. This approach also captured overhead scenarios relevant to multi-tenant environments or remote users experiencing intermittent connectivity.

File sizes ranged from 1 KB to 10 GB, reflecting variations from small overhead-dominated transfers to large, sustained throughput operations. Each file was copied multiple times: as many as 1,000 times for smaller files (1 KB to 10 MB) and fewer iterations (100 or 10) for larger files (10 MB to 10 GB). This approach maintained stable averages without excessively extending test durations. Throughput was recorded using a high-resolution .NET

timer, as it was found that built-in PowerShell 'Measure-Command' cmdlet was not precise enough for small files. CPU usage was collected through performance counters on both the client and the server, and memory usage was measured by recording the baseline free memory at the start of each run and noting any decreases observed during file copies, thereby capturing the delta in available RAM. To measure CPU usage accurately, emphasis was placed on the System process (PID 4), which is primarily responsible for file transfer operations on both client and server endpoints. Preliminary checks showed that the svchost process hosting the Workstation service consumed negligible CPU cycles, making it irrelevant for further measurement. Each run (e.g., 1000, 100, or 10 copies) was repeated five times under identical conditions, and min, max, and average metrics were aggregated.

C. Data Collection process

Four PowerShell scripts were employed to automate file transfers and record performance data. Two scripts tested SMB over TCP, while the other two tested SMB over QUIC. Each protocol was examined in both persistent version (single SMB mapping) and ephemeral version (mapping removed after each file), allowing comparisons of how session creation frequency influenced throughput and resource usage, especially for small files. The primary command for file copying was Robocopy, invoked with the /J (unbuffered) parameter to reduce caching effects. Precise timing of each operation was obtained using .NET's [QueryPerformanceCounter]. Windows' built-in performance counters (via logman.exe and Performance Monitor, PerfMon) were employed to record and display CPU usage, memory utilization, and related system metrics. In addition to monitoring CPU usage for the System process and Available MB counters, Bytes Sent/sec and Bytes Received/sec were also captured at the network interface. This approach ensured that CPU and memory measurements aligned with each script's execution timeline. Because PerfMon aggregates counters in a manner that can complicate detection of short-lived overhead, it is recommended that Windows Performance Analyzer (WPA) is used in future work. WPA provides more detailed tracing of process and thread events, making it easier to identify exactly where CPU time is being used. This is especially useful for analyzing ephemeral session overhead or cryptographic processing.

III. Results and Analysis

In the tables and charts below, throughput (in KB/s), CPU usage (in %), and memory usage (in MB) are presented for each test scenario.

TABLE I. Throughput in KB/s

File Size	TCP ephemeral session	TCP persistent session	QUIC ephemeral session	QUIC persistent session
1 KB	26,2698	40,8852	24,5028	36,7648
10 KB	268,8048	395,2604	244,158	359,7852
100 KB	2662,1234	4078,9366	2410,0126	3417,7344
1 MB	25213,8954	34566,2428	20819,7066	27145,6314
10 MB	220681,6968	287770,664	146471,0894	183313,0922
100 MB	1097756,39	1313158,919	353036,7488	392389,5612
1 GB	1306148,146	1286584,676	453773,0852	489448,5302
10 GB	482010,3618	620155,1496	403785,9756	404552,4504

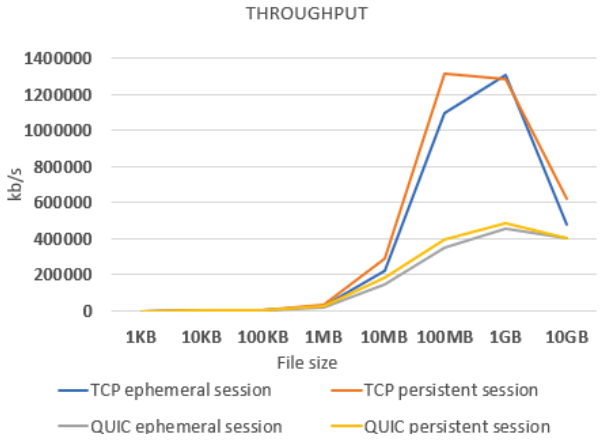

Figure 2. Average throughput on TCP and QUIC file transfer

TABLE II. CPU usage % on Client

File Size	TCP ephemeral session	TCP persistent session	QUIC ephemeral session	QUIC persistent session
1 KB	5,0352	2,8347	5,3124	5,318
10 KB	4,0858	2,862	4,7228	6,2674
100 KB	3,768	3,772	7,2132	6,626
1 MB	4,1004	3,1262	8,2098	12,8392
10 MB	7,1636	5,6706	17,5604	23,7178
100 MB	8,044	9,1354	29,2122	31,9932
1 GB	15,5798	17,8846	38,9612	44,3612
10 GB	13,8074	16,2848	43,3316	48,2716

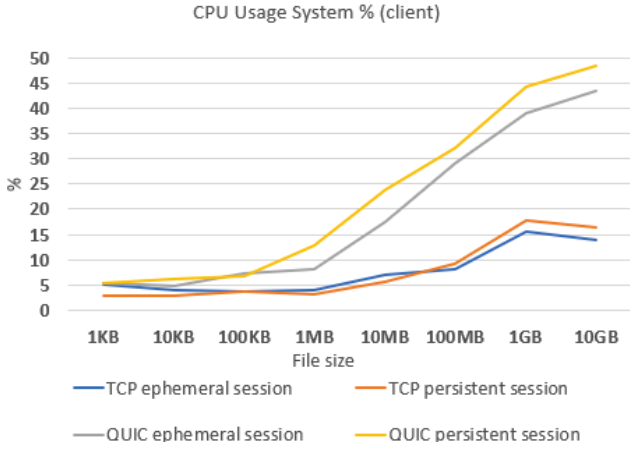

Figure 3. CPU Usage on Client

TABLE III. CPU USAGE % ON SERVER

File Size	TCP ephemeral session	TCP persistent session	QUIC ephemeral session	QUIC persistent session
1 KB	5,0352	2,8347	5,3124	5,318
10 KB	4,0858	2,862	4,7228	6,2674
100 KB	3,768	3,772	7,2132	6,626
1 MB	4,1004	3,1262	8,2098	12,8392
10 MB	7,1636	5,6706	17,5604	23,7178
100 MB	8,044	9,1354	29,2122	31,9932
1 GB	15,5798	17,8846	38,9612	44,3612
10 GB	13,8074	16,2848	43,3316	48,2716

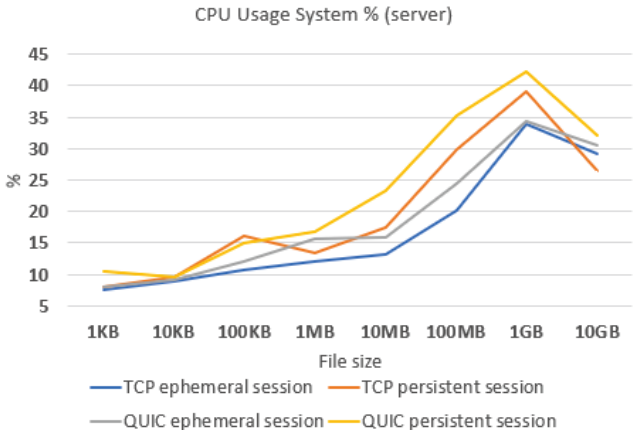

Figure 4. CPU Usage on Server

TABLE IV. MEMORY USAGE ON CLIENT

File Size	TCP ephemeral session	TCP persistent session	QUIC ephemeral session	QUIC persistent session
1 KB	191	183,4	142,4	213,4
10 KB	153,2	222,2	144,2	204,2
100 KB	155	181,4	147,2	198,4
1 MB	152,2	173,8	139,2	179
10 MB	143,6	190	104	184
100 MB	28,4	33,2	41,8	48
1 GB	25	23,6	40	40,6
10 GB	40,2	42	31,6	52,6

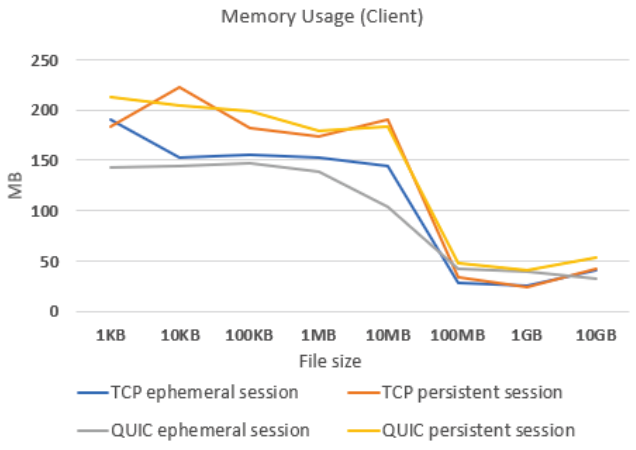

Figure 5. Memory Usage on Client

TABLE V. MEMORY USAGE ON SERVER

File Size	TCP ephemeral session	TCP persistent session	QUIC ephemeral session	QUIC persistent session
1 KB	55,2	48	51,2	44,4
10 KB	51,6	45,8	46,8	45,2
100 KB	54,4	66,2	72,4	85,8
1 MB	46,4	52,6	52,8	51,4
10 MB	44,4	49,8	45,2	52,2
100 MB	45,2	45,6	51,6	47,6
1 GB	51	47,4	49,6	50,4
10 GB	90	65	86	71,6

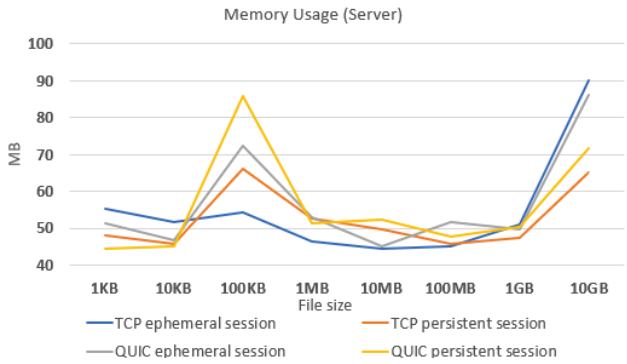

Figure 6. Memory Usage on Server

A. *Small Files (1 KB–100 KB)*

When transferring very small files, SMB's multi-step negotiation, authentication, and session creation tend to dominate the nominal data transfer. In ephemeral mode (where each file transfer includes mapping and unmapping), throughput sometimes dropped to 20–40 KB/s for 1 KB files, as session setup overshadowed the transfer of the negligible payload. Persistent connections reached hundreds of KB/s, although SMB's overhead remained substantial at this scale.

Packet-level traces indicated that SMB over QUIC might require four to six UDP datagrams for the QUIC handshake alone. Observations in Wireshark during a single file transfer showed that four QUIC packets were exchanged between client and server. SMB negotiation messages (Dialect Negotiate, Session Setup, Tree Connect) can add eight to ten additional datagrams, potentially exceeding 20 or 30 total datagrams in ephemeral mode for a 1 KB file. In one Wireshark capture, three packets were used for Negotiation, three for Session Setup, and two for Tree Connect. Overall, 70 packets were observed for the 1 KB transfer, a total that included multiple QUIC keep-alive packets.

A similar sequence of steps was observed for SMB over TCP, including the three-way handshake and multiple negotiation packets. Consequently, ephemeral small-file performance declined significantly for both TCP and QUIC. CPU usage for small files was minimal overall, with short spikes during session negotiation. Memory usage showed no significant distinction between TCP and QUIC aside from minor fluctuations.

B. Medium Files (1 MB–10 MB)

As file size increases, the ratio of overhead to actual transfer diminishes. For 1 MB, ephemeral overhead remained present, yet persistent sessions were able to reach tens of MB/s. TCP throughput approached 25–35 MB/s, whereas QUIC remained in the 20–27 MB/s range in the absence of packet loss or latency. The overhead of user-space encryption in QUIC contributed to additional CPU usage, but the limited data size allowed these transfers to complete quickly.

For 10 MB files, TCP again achieved higher raw throughput than QUIC, reinforcing the notion that kernel-based TCP leverages hardware offloads more efficiently in a near-zero-latency environment. Currently there is no hardware offload support for QUIC which also contributes to observed performance. [11] Packet traces revealed hundreds of segments or datagrams for 10 MB transfers, though ephemeral overhead receded after initial negotiation. Under persistent connections, the majority of traffic involved SMB write requests and responses.

C. Large Files (100 MB–1 GB–10 GB)

In large-file transfers, it was consistently observed that TCP could exceed 1 GB/s for 100 MB files, whereas QUIC typically remained between 300 MB/s and 600 MB/s. Ephemeral overhead became relatively minor at these sizes, and resource usage—especially CPU consumption—was identified as the key factor. A significant finding was that some 1 GB transfers attained higher average throughput than 10 GB transfers, as the Hyper-V host CPU was observed to reach 100% utilization for extended durations during 10 GB copies, resulting in throttled performance. Shorter transfers maintained higher burst rates before encountering thermal or resource constraints. Under local conditions without latency or packet loss, throughput depended heavily on CPU capacity for tasks such as cryptographic operations, bridging in the virtual switch, and user-space encryption overhead in QUIC. Kernel-level TCP, benefiting from more complete offloads, sustained near-peak speeds during long copies.

Despite the large-file throughput differences, an asymmetry in resource usage also emerged. The server's memory usage remained relatively stable, reflecting predictable caching in a dedicated file-server role. The client's memory usage, however, exhibited more fluctuation, potentially due to local caching, script overhead, and transient processes. CPU utilization also appeared consistently higher on the server side, particularly under encryption loads or repeated SMB session negotiations, suggesting that most disk I/O, cryptographic routines and authentication overhead fell on the server rather than the client.

IV. CONCLUSION

It has been shown in a low-latency Hyper-V setting that SMB over QUIC does not replicate the performance gains frequently associated with QUIC-over-HTTPS. Under HTTP/3, QUIC often surpasses TCP for small or short-lived requests thanks to features such as 0RTT and no head-of-line blocking. [1][4] However, SMB involves a heavier protocol stack with additional round trips for negotiation, share mapping, and file operations. Although QUIC's handshake may be streamlined, the higher-layer SMB interactions neutralize those benefits for smaller files. For large files in a near-perfect environment, the minimal latency, lack of packet loss and kernel-level TCP optimizations further reduce QUIC's usual advantages, clarifying why SMB over QUIC does not match QUIC-over-HTTPS performance. Under prolonged multi-gigabyte file transfers, the Hyper-V host's CPU typically becomes the limiting factor, especially with repeated cryptographic operations. While smaller (1 GB) transfers can complete quickly at near-peak speed, longer (10 GB) transfers trigger CPU saturation or thermal throttling, thereby reducing overall throughput.

Despite these observations, SMB over QUIC offers appealing properties for secure, firewall-friendly access and might demonstrate stronger comparative performance in real WAN scenarios, where moderate latency or loss undermines TCP's efficiency. Further kernel-level or offloaded encryption optimizations for SMB over QUIC may enhance performance and narrow the gap with TCP [11]. Ultimately, while QUIC consistently provides benefits in HTTP/3 across a range of network conditions, the inherent complexity of SMB constrains QUIC's effectiveness in a lab setting free from latency or loss, leaving TCP as the higher-performing option for bulk file transfers.

V. FUTURE WORK

As described in Section 2, the test environment was constructed in a workgroup setting without DNS or Active Directory infrastructure. Future investigations could incorporate domain-based authentication or multiple concurrent connections to obtain additional insights into SMB over QUIC performance. Researchers are also encouraged to introduce synthetic latency at the virtual network adapter level or to inject packet loss by interposing an additional VM as a router, thereby simulating WAN conditions where TCP experiences more frequent retransmissions.[6][12]

REFERENCES

[1] P. Kumar, "QUIC (Quick UDP Internet Connections) - A Quick Study", Santa Clara University, 2020, doi: 10.48550/arXiv.2010.03059

[2] P. N. N. G et al., "A Detail Survey on QUIC and its Impact on Network Data Transmission," 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2022, pp. 378-385, doi:10.1109/ICOEI53556.2022.9777199

[3] S. Cook, B. Mathieu, P. Truong and I. Hamchaoui, "QUIC: Better for what and for whom?", 2017 IEEE International Conference on Communications (ICC), Paris, France, 2017, pp. 1-6, doi:10.1109/ICC.2017.7997281

[4] O. Nalawade, A. Dhanwani and T. Prabhu, "Comparison of Present-day Transport Layer network Protocols and Google's QUIC," 2018 International Conference on Smart City and Emerging Technology (ICSCET), Mumbai, India, 2018, pp. 1-8, doi: 10.1109/ICSCET.2018.8537265

[5] K. Nepomuceno, et al., "QUIC and TCP: A Performance Evaluation", 2018 IEEE Symposium on Computers and Communications (ISCC), 2018, doi:10.1109/ISCC.2018.8538687

[6] X.Zhang, et al., “QUIC is not Quick Enough over Fast Internet”, doi:10.48550/arXiv.2310.09423

[7] J.Mücke, M.Nawrocki, R.Hiesgen, T.C.Schmidt, M.Wählisch.: “ReACKed QUICer: Measuring the Performance of Instant Acknowledgments in QUIC Handshakes”, ACM Internet Measurement Conference (IMC), 2024, doi:10.1145/3646547.3689022

[8] Microsoft Learn, 2025, “SMB over QUIC”, https://learn.microsoft.com/en-us/windows-server/storage/file-server/smb-over-quic?tabs=windows-admin-center%2Cpowershell2%2Cwindows-admin-center1 , accessed: December 2024

[9] Microsoft Learn, 2025, “Set-SMBClientConfiguration”, https://learn.microsoft.com/en-us/powershell/module/smbshare/set-smbclientconfiguration?view=windowsserver2025-ps, accessed: November 2024

[10] Wireshark with Custom Frida Python Script (2023), [Decrypting Schannel TLS traffic], https://b.poc.fun/decrypting-schannel-tls-part-1/#7-putting-it-all-together, accessed: December 2024

[11] X.Yang, L.Eggert, J.Ott, S.Uhlig, Z.Sun, G.Antichi "Making QUIC Quicker With NIC Offload", 2024 Usenix Annual Technical Conference, doi:10.1145/3405796.3405827

[12] A.Buchet, C.Pelsser, "An Analysis of QUIC Connection Migration in the Wild", 2024, doi:10.48550/arXiv.2410.06066

[13] J. Iyengar, M.Thomson, QUIC: A UDP-Based Multiplexed and Secure Transport, RFC 9000, 2021, doi:10.17487/RFC9000