# MQTT over QUIC: Comparison with TCP and TCP+TLS in a Virtualized IoT Environment

Davi S. de Luna
*Campus Igarassu*
*Instituto Federal de Pernambuco*
Igarassu, PE, Brazil
dsl6@discente.ifpe.edu.br

Ramon M. S. Farias
*Campus Igarassu*
*Instituto Federal de Pernambuco*
Igarassu, PE, Brazil
ramon.farias@igarassu.ifpe.edu.br

David J. M. Cavalcanti
*Centro de Informática*
*Universidade Federal de Pernambuco*
Recife, PE, Brazil
djmc@cin.ufpe.br

***Abstract*—IoT applications require reliable, efficient, and secure communication due to their distributed and resource-constrained nature. Message Queuing Telemetry Transport (MQTT) is widely adopted in these environments for its lightweight design, requiring minimal resources and optimizing network bandwidth. However, it still faces performance issues due to TCP and Transport Layer Security (TLS) overhead. Quick UDP Internet Connections (QUIC) has emerged as a promising alternative to improve communication performance. However, existing performance studies lack evaluations of MQTT over QUIC under low latency and unstable network conditions. This paper compares the performance of MQTT over QUIC, TCP, and TCP+TLS in a virtualized IoT environment. The performance evaluation follows a well-established approach and analyses latency and connection establishment time as key metrics under stable and unstable conditions. Results highlight the potential of QUIC as an efficient and secure alternative for IoT communication.**

***Index Terms*—Performance Evaluation, IoT, MQTT, QUIC, TCP**

## I. Introduction

The Internet of Things (IoT) is a technological paradigm that enables the development of systems based on a network of distributed smart devices, such as sensors, actuators, vehicles, and virtual entities. These devices work together over the Internet to provide IoT applications in diverse domains, such as smart cities [19] and Industrial IoT (IIoT) [15]. As IoT applications continue to evolve in complexity and connectivity, new challenges arise, including communication protocol design, technological disparities, and managing large volumes of data with limited resources. These challenges include the need to meet strict Quality of Service (QoS) requirements, such as efficiency, security, low energy consumption, and reliable communication, which rely on low latency, minimal packet loss, and other performance metrics [7], [11].

In this context, the Message Queuing Telemetry Transport [2] (MQTT) protocol has been widely adopted in IoT applications due to its lightweight nature and efficiency with resource-constrained devices and intermittent connectivity [13]. However, it still presents certain performance limitations. Since it operates over the TCP transport protocol, MQTT is subject to latency-related challenges such as the Head-of-Line Blocking (HOLB) phenomenon [12] and the overhead of error control in wireless networks, which can negatively affect performance [14]. Additionally, while using Transport Layer Security (TLS) is essential for securing MQTT communications, it introduces extra latency, making it less suitable for latency-sensitive IoT applications. Vieira et al. [18] evaluated the performance and energy efficiency of MQTT, MQTT+TLS, and CoAP in IoT environments, highlighting how protocol selection significantly influences time-sensitive applications. These insights motivate exploring alternative protocols, such as QUIC, which integrates TLS 1.3 natively and avoids TCP's Head-of-Line Blocking.

The Quick UDP Internet Connections [5] (QUIC) transport protocol, which is based on UDP, emerges as a promising approach to addressing these limitations in IoT applications. It removes HOLB and natively incorporates TLS 1.3, providing an integrated security solution while reducing the time required to establish secure connections [12]. Although QUIC is gaining increasing adoption [16], there are still few studies that explore its potential as an alternative to TCP, particularly in the context of MQTT and its performance in IoT applications, which motivates this research.

In this context, this paper presents a performance evaluation of MQTT using the QUIC protocol in a virtualized IoT environment, comparing it with traditional approaches such as TCP and TCP+TLS. In particular, the evaluation considers different MQTT implementations over these transport protocols and analyses key metrics such as latency (round-trip time) and connection establishment time, following the well-established performance evaluation approach proposed by Raj Jain [6]. The main goals are to assess MQTT over QUIC and provide results that support the development of IoT applications, guiding the selection of more suitable protocols for latency-sensitive applications requiring security. The findings serve as a reference for future large-scale validations and in physical test environments. It is worth noting that the benefits of a virtualized IoT environment are enabling reproducible, systematic, and parametrized evaluations and isolating real-world influences.

The rest of this paper is organized as follows: Section II presents related works. Section III presents the evaluated protocols. Section IV details the performance evaluation. Section V discusses the results. Finally, Section VI concludes the paper and outlines directions for future work.

## II. Related Work

This section reviews existing research on MQTT and QUIC in IoT environments, focusing on performance evaluation, security, and suitability for low-latency IoT applications.

Kumar and Dezfouli [9] implemented and analyzed the use of QUIC in IoT environments, demonstrating that this protocol can significantly reduce connection and response times compared to the traditional TCP+TLS stack. The main advantage observed in their work is the mitigation of the Head-of-Line Blocking issue inherent to TCP, along with the native integration of security through TLS 1.3. However, their study focuses on response time rather than latency. In contrast, the present work analyzes latency and connection establishment time, which are crucial in IoT applications where devices operate under dynamic and resource-constrained conditions and require low latency for efficient transmission.

Similarly to this performance evaluation, Fernández et al. [3] evaluated the performance of MQTT over QUIC in IoT environments. Their results indicate that QUIC can reduce connection establishment time and latency compared to traditional TCP+TLS implementations. Their findings reinforce the benefits of QUIC's ability to mitigate Head-of-Line Blocking while natively integrating security.

Iqbal et al. [4] investigated the performance of the AMQP protocol over QUIC in IoT networks. Their study showed that implementing QUIC can improve communication efficiency, particularly in resource-constrained environments requiring low-latency communication. While their research demonstrated QUIC's advantages for AMQP, the present work expands on this line of research by directly analyzing MQTT over QUIC and comparing it to the conventional MQTT/TCP/TLS stack.

Regarding security limitations in IoT environments, Alharbi and Aspinall [1] highlighted the need for protocols that provide efficient encryption without negatively impacting latency. In this context, the use of QUIC as a transport layer protocol for MQTT appears as a promising alternative, as it enables better performance and data security in an integrated manner when compared to TCP and TCP+TLS stacks. In this sense, this work empirically evaluates and provides concrete data on how QUIC's security integration affects latency and connection establishment time.

Zhang et al. [20] conducted a systematic evaluation of QUIC's performance over high-speed networks, revealing that the UDP+QUIC+HTTP/3 stack can experience up to a 45.2% reduction in data rate compared to the traditional TCP+TLS+HTTP/2 stack. This performance degradation was primarily attributed to high receiver-side processing overhead, notably due to excessive data packets and QUIC's user-space acknowledgment mechanisms. Although their study focused on web applications, such as file transfers and video streaming, their findings underscore the necessity of assessing QUIC's performance across various network contexts. In this regard, this study complements this analysis by evaluating the application of QUIC in MQTT-based communication within virtualized environments that simulate IoT scenarios, comparing its performance with MQTT/TCP and MQTT/TCP+TLS stacks in terms of latency and connection establishment time.

In summary, this paper differentiates itself by conducting experimental evaluations that directly compare the performance of the MQTT/TCP+TLS and MQTT/QUIC stacks in a controlled environment, considering metrics such as latency (publish time) and connection establishment time. This approach aims to provide concrete data on the benefits and challenges of using QUIC in IoT environments, contributing to the advancement of research in the field and offering insights for applications that require low latency and data protection.

## III. Background

This section provides a brief overview of the protocols evaluated in this paper: MQTT, TCP, TCP+TLS, and QUIC. MQTT is the *de facto* standard application-layer communication protocol for IoT systems [13]. TCP and TCP+TLS are widely adopted transport-layer protocols, with TLS providing secure communication. In contrast, QUIC introduces key innovations that directly address core IoT communication challenges, such as reducing latency, minimizing connection overhead, and ensuring secure data transmission.

### A. MQTT and IoT

MQTT (Message Queuing Telemetry Transport) [2] is an application-level messaging protocol widely adopted in IoT environments. Its lightweight design and small code footprint make it ideal for resource-constrained devices. It supports loosely coupled and asynchronous communication, enabling efficient message exchange from one to many devices [2].

MQTT follows a publish/subscribe communication model. Its architecture comprises three core components: publishers, a broker, and subscribers. Publishers are applications on devices that produce and send data to the broker. The broker is an intermediary, managing subscriptions and forwarding data to the appropriate subscribers. Subscribers are applications that register interest in receiving specific data.

While MQTT is commonly implemented over TCP, a variant called MQTT-SN (for Sensor Networks) uses UDP to enable faster data transmission in constrained networks. Additionally, integrating MQTT with QUIC may further enhance performance and security in IoT systems [4].

### B. TCP and TCP+TLS

TCP (Transmission Control Protocol) is a connection-oriented transport protocol that ensures reliable, in-order delivery of data through features like flow control, congestion control, and retransmissions. However, it suffers from Head-of-Line Blocking (HOLB), where the delay of one packet stalls all subsequent packets in a stream [12].

When combined with TLS (Transport Layer Security), as in MQTT/TCP+TLS, an additional handshake is required before data can be securely exchanged. This adds latency to the connection setup process, which can significantly impact time-sensitive IoT applications. TLS also increases computational overhead due to encryption and authentication mechanisms.

Another important limitation of TCP and TCP+TLS is their poor handling of mobility. They tightly couple the connection state to the client's IP address. As a result, when a device changes networks, for example, switching from Wi-Fi to cellular, the existing connection is dropped, requiring the entire setup process (including TCP and TLS handshakes) to be restarted [9]. This behavior creates challenges in IoT scenarios characterized by mobility or frequent network changes.

### C. QUIC

QUIC (Quick UDP Internet Connections) is a transport protocol developed by Google and later standardized by the IETF RFC 9000 [5]. Unlike TCP, QUIC operates over UDP and integrates TLS 1.3 as a mandatory component of the protocol, eliminating the need for a separate security handshake.

QUIC addresses the limitations of TCP in several ways [5]:

- **Low-Latency Handshake:** QUIC enables 0-RTT and 1-RTT connection establishment, significantly reducing the time needed to initiate secure communication.
- **Multiplexing Without HOLB:** QUIC supports multiple streams within a single connection without HOLB, even in the presence of packet loss.
- **User-Space Implementation:** QUIC implementations usually reside in user space, allowing rapid protocol evolution and optimization across platforms.
- **Integrated Security:** TLS 1.3 is built into the QUIC handshake [17], ensuring confidentiality and authentication from the start.

These characteristics make QUIC particularly appealing for IoT environments, where connection efficiency and security are critical.

There is a difference between TCP+TLS and QUIC handshakes for establishing secure communication. While TCP+TLS requires multiple round-trips, first for the TCP 3-Way Handshake, then for the TLS 1.2 Handshake to establish a secure session, QUIC streamlines this process efficiently by combining transport and security negotiation (TLS 1.3 with ECDHE) into a single exchange. This enables 1-RTT for session establishment and 0-RTT resumption, making it especially advantageous in constrained or unstable IoT applications.

## IV. Performance Evaluation

Following the approach proposed by Jain [6], multiple experiments were conducted to evaluate the MQTT over QUIC, TCP and TCP+TLS transport protocols. These evaluations have two main objectives: to measure the impact of QUIC on the performance of the MQTT protocol and to compare its performance with TCP and TCP+TLS in the same scenarios.

### A. Experiment setup

Fig. 1 presents the evaluation scenario.

To evaluate the MQTT over the QUIC protocol and compare it with TCP and TCP+TLS, an IoT publisher Application was implemented using MQTT over each transport protocol. This publisher automatically generates computation job messages published at configurable time intervals to the Broker. In turn, the subscriber application, Jobs Computer, subscribes to a specific job topic to receive these computation jobs, processes them, and publishes the results to a result topic to which the application has previously subscribed.

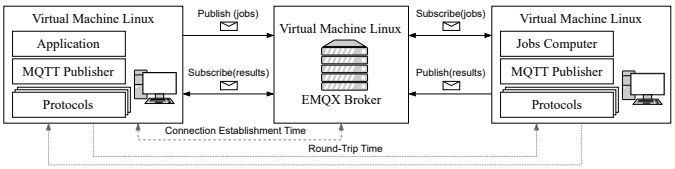


Fig. 1. Overview of the Evaluation Scenario

It is worth noting that a job refers to any computational activity (e.g., sending a message or performing a mathematical operation) performed by applications. MQTT has been widely adopted as a standard protocol for implementing publish/subscribe systems in IoT environments. For example, vehicular cloud computing leverages underutilized resources, such as computing power and communication capabilities, to form a cloud infrastructure that applications can use to process their jobs.

Fig. 2 summarizes the protocol configurations used during the experiments. At the Application layer the MQTT protocol was used to enable communication between the application and the job computer. Initially, it was combined with the TCP protocol and with the TCP+TLS stack. Then, MQTT was used over QUIC to analyse and compare it with the other configurations. These combinations were chosen to compare the expected communication stack (MQTT over QUIC) with the traditional ones (MQTT over TCP and MQTT over TCP+TLS). In particular, the TLS layer is added over TCP to ensure secure communication, while QUIC natively integrates TLS 1.3, providing security without the need for an additional layer.

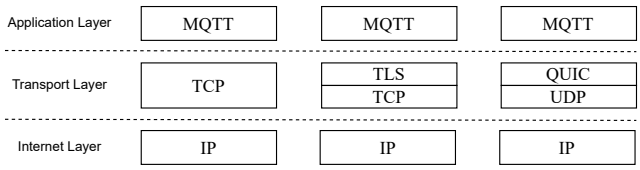


Fig. 2. Protocol Scenarios

In the evaluation scenario, the Publisher, Subscriber and the Broker were executed on three virtual machines (VMs) configured to simulate a realistic network environment and measure the latency and connection establishment time between the Publisher and Subscriber, and the Publisher and the Broker, respectively. The VMs were setup to emulate IoT devices (e.g., Raspberry Pi), each with 2 GB of RAM, a dual-core processor, and a network interface in Host-only mode, ensuring isolated and controlled communication. The operating system used in the VMs was Ubuntu 22.04.4 LTS. The virtualization software was VirtualBox version 7.0.20 r163906 (Qt5.15.2).

All components used in the experiments were deployed and executed on a PC equipped with an Intel(R) Core(TM) i7-

10700 CPU at 2.90GHz, 32 GB of DDR4 RAM, and running on Windows 11 Pro 64-bit, version 24H2.

### B. Metrics, parameters and factors

In all experiments, two metrics were used to evaluate performance: the *Connection Establishment Time* ($T_{\text{Connection}}$) and the *Round-Trip Time* (RTT). The ($T_{\text{Connection}}$) measures the time required for the Publisher to establish a connection with the Broker. It is calculated as the difference between the timestamp at the start of the connection process, when the CONNECT packet is sent ($T_{\text{Publisher\_req}}$) to the Broker, and the timestamp of the Broker's response ($T_{\text{Broker\_ack}}$), as shown in Eq. 1.

$$T_{\text{Connection}} = T_{\text{Broker\_ack}} - T_{\text{Publisher\_req}} \tag{1}$$

RTT is measured on the Publisher side. It represents the time elapsed from when the Publisher sends a PUBLISH message with a job to the Broker ($T_{\text{Send}}$) until it receives the job results sent by the Subscriber ($T_{\text{Received}}$), as shown in Eq. 2.

$$RTT = T_{\text{Received}} - T_{\text{Send}} \tag{2}$$

The following workload parameters were considered in the experiments: three different configurations of MQTT Publisher using MQTT over QUIC (MQTT/QUIC), MQTT over TCP (MQTT/TCP) and MQTT over TCP+TLS (MQTT/TCP+TLS). In addition, the evaluation included different message publication intervals, set at 50 and 100 ms. Furthermore, the experiments also adopted MQTT with QoS level 0 (Fire and Forget). The $tc$ tool was used on the network interface with two configurations: without packet loss and introducing a packet loss rate of 20% to simulate both stable and unstable network conditions. This choice follows the approach of [10], which emulates typical IoT environments with wireless communication prone to loss, delay, and external disruptions. Finally, the connection establishment operation was repeated 100 times, and the publish operation was repeated 1000 times in each experiment.

Table I summarizes the parameters and their respective factors utilized in all the experiments.

TABLE I
SUMMARY OF PARAMETERS AND THEIR RESPECTIVE FACTORS IN THE EXPERIMENTS

| **Parameter** | **Factors** |
|---|---|
| Transport Protocol | MQTT/QUIC, MQTT/TCP, MQTT/TCP+TLS |
| Publication Interval | 50 ms, 100 ms |
| Packet Loss Rate | 0%, 20% |
| MQTT QoS Level | 0 (Fire and Forget) |
| Publish repetition | 1000 times |
| Connection repetition | 100 times |

### C. Implementation

As for the implementation[1], elements such as the Application and Job Computer were implemented using C, and the NanoSDK was used to implement the MQTT protocols. EMQX was used as the Broker. Data storage was handled using Redis, with the Hiredis library for efficient communication and processing. Data collection was performed using tcpdump through packet capture.

## V. Results and Discussion

This section discusses the performance evaluation results[2] obtained from the collected data. The metrics were measured by recording timestamps at message sending and receiving events. The stored data were analyzed through descriptive statistics and comparative analysis between the protocols. Finally, the average *Connection Establishment Time* to the Broker and average *RTT* were considered.

### A. Impact of QUIC on the performance of MQTT

This evaluation aims to analyze the impact of QUIC on MQTT, focusing on the *Connection Establishment Time*, an essential metric given the dynamic behavior of IoT devices, which frequently connect and disconnect. The experiments were conducted under two network conditions: a stable environment, without packet loss, and another unstable, with 20% packet loss. Finally, the average connection establishment time was calculated based on 100 operations.

Figure 3 presents the average *Connection Establishment Time* for MQTT running over TCP, TCP+TLS, and QUIC in stable and unstable network conditions. This comparison makes it possible to assess whether QUIC improves MQTT connection establishment, especially in scenarios with packet loss, where handshake performance becomes needed.

In the stable condition (0% packet loss), QUIC had the best *Connection Establishment Time*, with an average of 0.0027 ms. TCP followed closely with 0.0029 ms, while TCP+TLS showed a significantly higher average of 0.0114 ms, which is expected due to the added overhead of the TLS handshake phase.

Under the unstable condition (20% packet loss), all protocols presented increased *Connection Establishment Time*, reflecting the additional network overhead. TCP rose to 0.3382 ms, TCP+TLS to 0.3356 ms, and QUIC to 0.1882 ms. Despite the degradation, QUIC remained the fastest, outperforming TCP by 44.32% and TCP+TLS by 43.92%. This shows that QUIC handles connection establishment more efficiently, even in adverse network conditions, benefiting from its integrated security and reduced latency requirement.

Overall, QUIC shows the most robust performance under packet loss, maintaining low connection latency while integrating security features natively. This resilience makes QUIC particularly suitable for dynamic IoT environments, where devices frequently connect and disconnect, and where packet loss is common due to wireless communication or network instability. In such scenarios, quickly establishing secure connections while preserving low latency is crucial for maintaining reliable, responsive application behavior.

[1]https://encurtador.com.br/xCRfA

[2]https://encurtador.com.br/ocQ6y

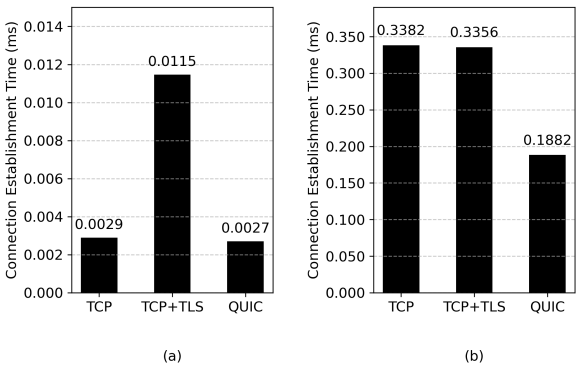


Fig. 3. *Connection Establishment Time* for MQTT over TCP, TCP+TLS, and QUIC. (a) No packet loss. (b) 20% packet loss.

### B. Comparative evaluation

This second experiment compares the performance of QUIC, TCP and TCP+TLS. The adopted metric is the *Round-Trip Time* (RTT), measured using two message publication intervals: 50 ms and 100 ms. In addition, two network conditions were considered: one with no packet loss (0%) and another with 20% packet loss.

Figure 4 shows the 50 ms publication interval results. Under ideal network conditions (no packet loss), TCP showed the best average *RTT* with approximately 0.0041 ms, followed closely by QUIC with 0.0050 ms and TCP+TLS showing a slightly higher value with 0.0051 ms. These values are very close, indicating that under low-latency, stable conditions, the impact of the underlying transport protocol is minimal for this interval.

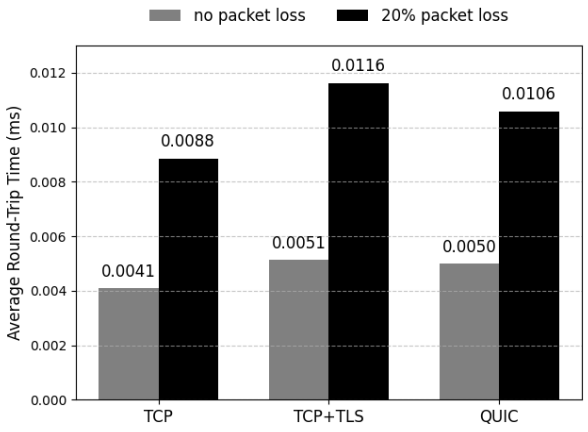


Fig. 4. RTT Comparison for MQTT over TCP, TCP+TLS, and QUIC (50 ms Interval)

However, when 20% packet loss was introduced, performance differences became more evident. TCP achieved the best performance, with an average RTT of 0.0088 ms. In comparison, QUIC recorded 0.0106 ms, and TCP+TLS, 0.0116 ms. This shows that, although QUIC was not the fastest, it handled security better under degraded conditions. It's also important to note that the relative increase in RTT for QUIC (112%) was smaller than that of TCP+TLS (127%), indicating slightly better resilience to loss despite not achieving the lowest absolute RTT.

Figure 5 shows the 100 ms publication interval results under both network conditions. As can be observed, for the 100 ms publication interval, TCP also had the lowest average *RTT* with 0.0042 ms without packet loss, followed by QUIC with an average *RTT* time of 0.0050 ms and TCP+TLS with 0.0055 ms. Under 20% packet loss, TCP presented a *RTT* of 0.0091 ms, while QUIC and TCP+TLS recorded 0.0118 ms and 0.0105 ms, respectively. Interestingly, TCP+TLS slightly outperformed QUIC in this condition, which contrasts with expectations since QUIC is typically designed to be more resilient to packet loss.

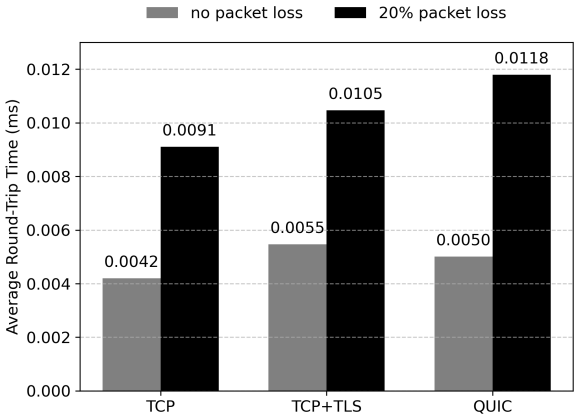


Fig. 5. RTT Comparison for MQTT over TCP, TCP+TLS, and QUIC (100 ms Interval)

As the publication interval increases, TCP tends to have the best raw performance, especially in networks without packet loss. However, QUIC consistently outperforms TCP+TLS and natively integrates TLS, providing built-in security capabilities. This makes QUIC a strong candidate for secure and low-latency communication in IoT.

To statistically support the observed differences in RTT, Kruskal–Wallis tests [8] were performed across all combinations of publication interval and network condition. This non-parametric test was chosen due to its robustness against violations of normality assumptions in latency distributions. These results[3]

In summary, TCP showed the most consistent raw RTT performance in both intervals and conditions, particularly in lossless scenarios. QUIC showed competitive results and clear advantages over TCP+TLS, making it a strong candidate for

[3]https://encurtador.com.br/hiRIq confirm that the differences in latency distributions across protocols are statistically significant under all tested conditions. This reinforces the importance of selecting appropriate transport protocols for real-time IoT applications, especially in lossy or unstable networks.

secure and low-latency IoT applications. Its benefits become more prominent in scenarios involving frequent connections or devices operating under lossy, constrained networks, conditions common in many real-world IoT deployments.

## VI. Conclusion and Future Work

This paper presented a performance evaluation of MQTT over the QUIC protocol, comparing it to the conventional TCP and TCP+TLS stacks for supporting MQTT. The evaluation considered different MQTT implementations over these transport protocols in a virtualized IoT environment . Key metrics such as *Round-Trip Time* (RTT) and *Connection Establishment Time* were analyzed, following Jain's performance evaluation approach.

The results showed that QUIC performed better in low-latency scenarios, reducing connection time and latency compared to TCP+TLS, while offering the advantage of natively integrated security and improved robustness under network impairments. However, this advantage was less evident in higher latency conditions, with TCP showing more stable performance. TCP+TLS introduced additional delays due to cryptographic negotiation, making it less suitable for latency-sensitive applications. In contrast, QUIC integrates TLS natively, reducing this overhead.

The experiments were conducted using VMs because the NanoSDK does not support the ARM architecture, preventing QUIC from running directly on IoT devices. However, virtual environments provide a controlled configuration to simulate devices, isolating real-world interference and ensuring that the experiments are systematized and parameterized, thereby facilitating replication and retesting. The RTT values obtained were considerably lower than those typically observed in physical IoT deployments, which should be taken into account when interpreting the results, as the virtualized environment in host-only mode may overestimate performance. Furthermore, the study did not address aspects such as reconnection, multiplexing, and interoperability with other protocols, which are important features of QUIC that can enhance performance in high-demand scenarios.

As for future work, implementing QUIC on physical IoT devices, such as Raspberry Pi, is essential to assess its real-world performance. Expanding the evaluation to include large-scale tests, more diverse network conditions, and varying workload would also provide a broader understanding of its behavior in complex scenarios. Another key direction to explore is QUIC's stream multiplexing, evaluating its impact on reconnection and high-demand environments. An unexpected result was observed where QUIC did not outperform TCP under certain packet loss conditions, despite being designed to handle such impairments more effectively. Investigating this behavior in greater depth, particularly the loss detection mechanisms of the QUIC protocol, could clarify these findings. Finally, studying QUIC's interoperability with other systems and protocols would help gauge its potential for broader adoption in heterogeneous IoT environments.

## References


[1] Rana Alharbi and David Aspinall. An iot analysis framework: An investigation of iot smart cameras' vulnerabilities. In *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, pages 1–10, March 2018.

[2] Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. Mqtt version 5.0. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, 2019. OASIS Standard.

[3] Fátima Fernández, Mihail Zverev, Pablo Garrido, José R. Juárez, Josu Bilbao, and Ramón Agüero. And quic meets iot: performance assessment of mqtt over quic. In *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–6, Oct 2020.

[4] Faheem Iqbal, Moneeb Gohar, Hani Alquhayz, Seok-Joo Koh, and Jin-Ghoo Choi. Performance evaluation of amqp over quic in the internet-of-thing networks. *Journal of King Saud University - Computer and Information Sciences*, 35(4):1–9, 2023.

[5] Jana Iyengar and Martin Thomson. Rfc 9000: Quic: A udp-based multiplexed and secure transport. *Omtermet Emgomeeromg Task Force*, 2021.

[6] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, New York, United State of America, 09 1991.

[7] Amir Javadpour, Guojun Wang, and Samira Rezaei. Resource management in a peer to peer cloud network for iot. *Wirel. Pers. Commun.*, 115(3):2471–2488, December 2020.

[8] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[9] Puneet Kumar and Behnam Dezfouli. Implementation and analysis of quic for mqtt. *Computer Networks*, 150:28–45, 2019.

[10] Elizabeth Liri, Prateek Kumar Singh, Abdulrahman BIN Rabiah, Koushik Kar, Kiran Makhijani, and KK Ramakrishnan. Robustness of iot application protocols to network impairments. In *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 97–103. IEEE, 2018.

[11] Eduardo Magrani. *A Internet of Things*. FGV Editora, Rio de Janeiro, 2018. Text in Portuguese.

[12] Michael Scharf and Sebastian Kiesel. Nxg03-5: Head-of-line blocking in tcp and sctp: Analysis and measurements. In *IEEE Globecom 2006*, pages 1–5, Nov 2006.

[13] Victor Seoane, Carlos Garcia-Rubio, Florina Almenares, and Celeste Campo. Performance evaluation of coap and mqtt with security support for iot environments. *Computer Networks*, 197:108338, 2021.

[14] Wentao Shang, Yingdi Yu, Ralph Droms, and Lixia Zhang. Challenges in iot networking via tcp/ip architecture. Technical Report NDN-0038, NDN Project, February 2016.

[15] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, Nov 2018.

[16] Erik Sy, Christian Burkert, Hannes Federrath, and Mathias Fischer. A quic look at web tracking. *Proceedings on Privacy Enhancing Technologies*, 2019:255–266, 07 2019.

[17] M Thomson and S Turner. Rfc 9001: Using tls to secure quic, 2021.

[18] Emanuel Vieira, Murilo Cervi, Renato Azevedo, and Tiago Rizzetti. Performance and energy efficiency analysis of mqtt and coap protocols in the iot context. In *Extended Proceedings of the XXIV Brazilian Symposium on Information and Computational Systems Security*, pages 321–327, Porto Alegre, RS, Brazil, 2024. SBC. Text in Portuguese.

[19] Mostafa Zaman, Nathan Puryear, Sherif Abdelwahed, and Nasibeh Zohrabi. A review of iot-based smart city development and management. *Smart Cities*, 7(3):1462–1501, 2024.

[20] 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 Web Conference 2024*, pages 2713–2722, 2024.