Accepted in IEEE INDISCON 2026

On the Performance of MQTT over QUIC vs. TCP-TLS in 5G Smart Grid Networks

Aman Kumar*, Kritika Patidar*, Anand Vikram Singh*, Samaresh Bera*, and Zahid Akhtar†
*Dept. of Computer Science and Engineering, Indian Institute of Technology Jammu, India
†Dept. of Electrical and Computer Engineering, State University of New York Polytechnic Institute, USA
Email: {2022ucs0079, 2022ucs0095, 2023pcs0132}@iitjammu.ac.in, s.bera.1989@ieee.org, akhtarz@sunypoly.edu

***Abstract*—Advancements in cellular communications enable network operators to deploy 5G networks tailored to specific use cases, spanning across healthcare, smart grid, and industrial applications. The message queuing telemetry transport (MQTT) protocol is widely used as a prominent application-layer protocol for smart grid applications, such as smart metering, demand response management, and energy monitoring and control. Existing MQTT studies use the TCP-TLS stack to ensure reliable, secure communication. However, the associated overhead in the TCP-TLS stack makes it challenging to use in smart grid communications, where latency, reliability, and security are the tightly-coupled key performance indicators. This paper investigates the performance of MQTT over QUIC as an alternative transport for 5G-enabled smart grid communications. We develop lightweight MQTT clients using the NNG and Go QUIC libraries and integrating them with an EMQX broker to support publish–subscribe smart grid communication. The evaluation is conducted on a physical private 5G network and Raspberry Pi-based devices used for smart grid applications. Experimental results from the testbed show that QUIC significantly reduces end-to-end latency compared to traditional TCP-TLS-based methods, demonstrating its robustness under real 5G network scheduling and control-plane behavior. Furthermore, QUIC outperforms TCP-TLS in terms of CPU and memory usage, owing to QUIC's optimized 1-RTT handshake and fully user-space architecture. These findings position QUIC as a viable and efficient transport alternative to TCP-TLS for MQTT, particularly for 5G-enabled smart grid communications.**

***Index Terms*—MQTT, Transport layer security, QUIC, Performance evaluation, Private 5G networks**

I. Introduction

The emergence of 5G and beyond networks enables network operators to deploy different use-cases, such as healthcare, smart grid, and industrial applications, beyond the traditional cellular communications [1], [2]. In smart grid communications, smart metering, demand response management, and energy monitoring and control are supported by 5G networks. Typically, these applications use Message Queuing Telemetry Transport (MQTT) as the application layer protocol for exchanging real-time information due to its lightweight publish and subscribe messaging feature. Traditionally, MQTT runs over TCP, with TLS layered atop, to provide encryption and secure transmission. While this setup ensures reliable and secure communication, it is not suitable for use in latency-critical smart grid applications. This is because of the significant handshaking overhead involved in TCP's connection establishment and head-of-line blocking, specifically, visible in wireless and mobile networks, such as 5G [3]. Therefore, the issues with TCP make the integration of smart grid applications with stringent quality-of-service requirements challenging in 5G networks.

The Quick UDP Internet Connections (QUIC), a new transport protocol built on UDP, was designed to address these shortcomings of TCP [4]. QUIC combines TLS 1.3 encryption directly into the protocol, supports multiplexed streams to avoid head-of-line blocking, and operates entirely in user-space for faster innovation and flexibility [4], [5]. Consequently, MQTT over QUIC makes the approach suitable for fast, reliable, and secure smart grid communications. While there exist a few works that focused on integrating MQTT with QUIC [6]–[8], they primarily focused on theoretical analysis and traditional IoT networks. And, they overlooked the performance trade-offs of QUIC in a 5G-enabled network for critical applications, such as smart grid.

To address the research gap in the literature, in this paper, we study the performance of MQTT running over QUIC in 5G environments, specifically, tailored for smart grid communications. We develop lightweight MQTT clients using NNG [9] and Go QUIC libraries [10] and integrate them with an EMQX broker to support publish-subscribe smart grid communication. For the experiment setup, we utilize a private 5G lab testbed available at IIT Jammu along with Raspberry Pi-based devices used for smart grid applications. We consider continuous and periodic traffic patterns using persistent and non-persistent connections, respectively, to model wide range of smart grid applications [11], [12]. The experiment results indicate that MQTT over QUIC is a viable solution for real-time smart grid communications with reduced latency and low-resource consumption than that of using the traditional MQTT over TCP-TLS.

The rest of the paper is organized as follows. Section II discusses the existing works that focused on MQTT over QUIC. Section III presents the detailed system setup of MQTT with QUIC in a private 5G network. Section IV presents the experiment results. Finally, Section V concludes the paper with future research directions.

II. Related Work and Research Gaps

In this section, we review the existing works that focused on using MQTT over TCP-TLS and QUIC for IoT applications in general. There have been studies on how MQTT performs over traditional TCP connections [13], [14]. Most of the studies primarily focused on evaluating latency and throughput under stable and ideal network conditions. However, these works often neglect important system-level performance metrics, such as CPU and memory usage, which are critical for resource-constrained devices used in IoT communications. To address the limitations of TCP-TLS-based approaches, researchers also explored QUIC as a transport protocol with MQTT [6]–[8]. QUIC's features like faster connection establishment, built-in TLS 1.3 encryption and better performance under packet loss suggest that it can significantly improve MQTT behavior. However, research on the adoption and evaluation of QUIC in MQTT-based communication remains limited. Specifically, empirical studies on MQTT over QUIC are still sparse. We discuss some of the existing works that are directly aligned with this work as follows.

Kumar and Dezfouli [6] studied the performance of MQTT using QUIC as a transport protocol. The authors showed that the CPU usage at the MQTT client can be reduced by 83% and the memory usage by 50% compared to the traditional TCP-TLS-based mechanisms. The study was conducted in a controlled environment where bandwidth and packet loss are not challenging, unlike real-world mobile networks, such as 5G. Similarly, the authors in [7] benchmarked the performance of MQTT over QUIC, observing that QUIC consumes less time for connection establishment, and makes it suitable for bandwidth-limited networks such as WiFi, 4G, and satellite networks. However, their study was based on synthetic device loads and lacked fine-grained system-level measurements like per-session CPU profiling and resource utilization.

Fernández et al. [8] provided an empirical characterization of MQTT over QUIC by measuring delay and energy consumption across various wireless technologies such as WiFi, 4G LTE, and satellite networks. They showed that MQTT over QUIC reduced delay and jitter without significantly increasing energy consumption. The authors considered the persistent modes for session establishment. However, both persistent and non-persistent connections are viable in IoT communications, specifically, in smart grid communications, such as energy forecasting and monitoring, demand response, and smart metering. Furthermore, the system-level CPU and memory profiling, critical for resource-constrained IoT-based smart grid deployments, are not studied.

In summary, we find the following limitations in the existing works:

- Most of the studies considered persistent connections, where a session is maintained for a long time. However, maintaining ideal session for periodic data transfer may not be viable for resource-constrained smart grid communications.
- Limited study on system-level resource usage alongside latency measurements. This is crucial as smart grid devices can be resource-constrained.
- To the best of our knowledge, none of the previous studies have built a native C-based MQTT-over-QUIC client using NNG for performance evaluation in a private 5G network testbed. Furthermore, applications to smart grid communications have not been studied.

This paper aims to fill these gaps by building a native MQTT-over-QUIC system using NNG, deploying it in a 5G network, and rigorously measuring latency, CPU usage, and memory consumption across persistent and non-persistent sessions. This approach provides a holistic view of the practical feasibility of QUIC for lightweight and resource-sensitive smart grid applications using 5G networks.

III. System Setup

Figure 1 shows the 5G service-based architecture with the placements of the MQTT publisher, broker, and subscriber. Inside the UE of the 5G network (enabled with sensors for smart grid applications), we place the MQTT publisher, whereas the MQTT broker and subscriber are placed inside the 5G user-plane function (UPF) and data network (DN), respectively. The network, transport, and application layers of the MQTT publisher, broker, and subscriber are shown in the figure. Thus, the setup reflects smart grid communications in a private 5G network. Figure 2 shows the 5G testbed, where the MQTT broker is placed inside the UPF.

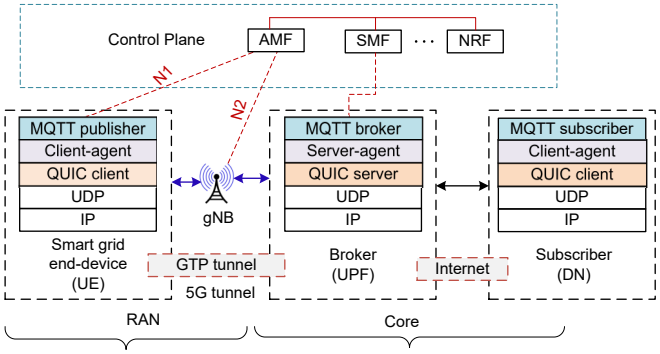


Fig. 1: Layered architecture of MQTT-QUIC and its placement in 5G testbed

On the other hand, the MQTT subscriber subscribes to data from the MQTT broker through a generic network interface. We discuss the implementation and working principles of each entity in subsequent sections.

1) MQTT Publisher (Raspberry Pi):

- Publisher is implemented on a Raspberry Pi running 64-bit OS. The Raspberry Pi is integrated with multiple sensors to enable energy monitoring in a smart grid.
- QUIC protocol at the publisher is implemented using Go and the `quic-go` library [10].
- The publisher connects to the 5G user-plane function (UPF) using the 5G tunnel interface. The traffic between

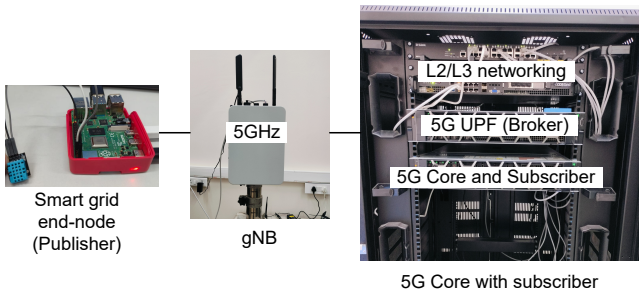


Fig. 2: 5G testbed with Raspberry-Pi for smart grid communications

the end-node and UPF is end-to-end encrypted using QUIC.

- The publisher for TCP/TLS-based transport is implemented using the Python Paho MQTT client [15].
- For persistent experiments, multiple publish operations are done using `CONNECT` → `PUBLISH messages` → `DISCONNECT` cycle.
- For non-persistent experiments, each publish operation follows a `CONNECT` → `PUBLISH` → `DISCONNECT` cycle.

The procedure for an MQTT publisher is as follows:

- `nng_mqtt_quic_client_open_conf()`: Initializes a QUIC client and prepares the connection configuration.
- `mqtt_msg_compose(PUB)`: Composes an MQTT PUBLISH message with payload and topic.
- `nng_sendmsg()`: Sends the composed message to the EMQX broker over QUIC.
- `nng_close()`: Closes the QUIC connection after transmission.

The connection setup and message transfer functionalities are as follows:

```
// Connection establishment
if (type == CONN) {
 nng_mqtt_msg_set_packet_type(msg,
     NNG_MQTT_CONNECT);
 nng_mqtt_msg_set_connect_proto_version(msg,
     4);
 nng_mqtt_msg_set_connect_keep_alive(msg, 30)
     ;
 nng_mqtt_msg_set_connect_clean_session(msg,
     true);
}
// Message transfer with broker
else if (type == PUB) {
 nng_mqtt_msg_set_packet_type(msg,
     NNG_MQTT_PUBLISH);
 nng_mqtt_msg_set_publish_dup(msg, 0);
 nng_mqtt_msg_set_publish_qos(msg, qos);
 nng_mqtt_msg_set_publish_retain(msg, 0);
 nng_mqtt_msg_set_publish_topic(msg, topic);
 nng_mqtt_msg_set_publish_payload
   (msg, (uint8_t *) payload, payload_size);
}
```

2) EMQX Broker:

- 5G UPF hosts the EMQX broker, configured to support MQTT over QUIC [16].
- The broker supports both MQTT over TCP/TLS (port 8883) and MQTT over QUIC (port 14567).
- It stores incoming publish messages and routes them to interested subscribers.
- It handles message routing and session-ticket generation for resumption.

3) MQTT Subscriber:

- The subscriber acts as an MQTT client for receiving messages from the broker.
- It is implemented using Go (QUIC) and Python Paho (TCP/TLS) similar to the MQTT publisher.
- It is hosted on a separate Ubuntu-based virtual machine inside the private 5G network.
- It subscribes to the test topic and records arrival timestamps.

The connection establishment and subscription of message from broker to subscriber functionalities are as follows:

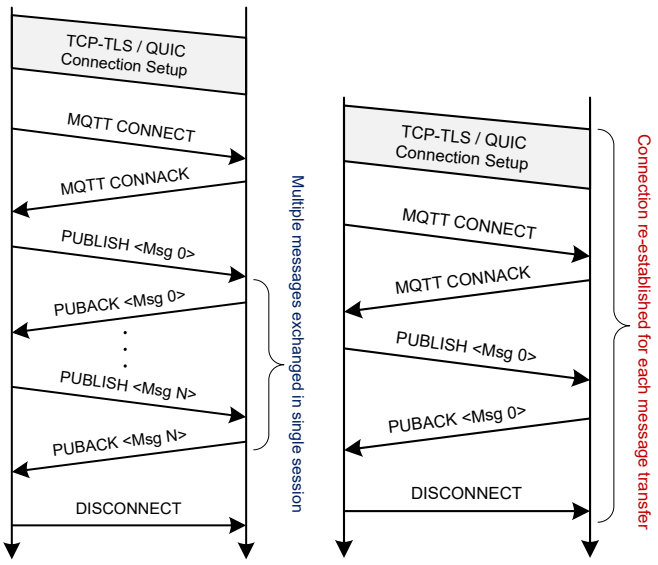
```
\\ Connection establishment
if (type == CONN) {
 nng_mqtt_msg_set_packet_type(msg,
     NNG_MQTT_CONNECT);

 nng_mqtt_msg_set_connect_proto_version(msg,
     4);
 nng_mqtt_msg_set_connect_keep_alive(msg, 30)
     ;
 nng_mqtt_msg_set_connect_clean_session(msg,
     false);
}

\\ Message subscription
else if (type == SUB) {
 nng_mqtt_msg_set_packet_type(msg,
     NNG_MQTT_SUBSCRIBE);
 nng_mqtt_topic_qos subscriptions[] = {
  {
   .qos   = qos,
   .topic = {
    .buf    = (uint8_t *) topic,
    .length = strlen(topic)
   }
  }
 }
}
```

IV. Performance Evaluation

We consider the following performance metrics to present a comparative analysis of MQTT over QUIC and TCP-TLS: latency in message transfer, CPU and memory resource utilization. We consider both persistent and non-persistent modes. Such consideration covers both continuous and periodic traffic in smart grid communications [11], [12]. In the persistent mode, a session is established initially, and multiple data messages are exchanged without closing the session, as shown in Figure 3(a). Whereas in non-persistent mode, a new session is established and closed between the MQTT publisher and broker for each periodic data transfer, as shown in Figure 3(b).



(a) Persistent connection (b) Non-persistent connection

Fig. 3: MQTT messages and data transfer

• **MQTT over TCP-TLS**: The following mathematical expression is used for calculating the session establishment time for MQTT with TCP-TLS:

$$L^{\text{np}}_{\text{tcp-tls}} = \sum_{i=0}^{N} \Big(\overbrace{T_{\text{tcp-tls}} + T_{\text{mqtt}}}^{\text{connection setup}} + \overbrace{S/B + D_{\text{prop}}}^{\text{transmission-propagation}} + \overbrace{L_{\text{proc}}}^{\text{processing}} \Big),$$

where $T_{\text{tcp-tls}} + T_{\text{mqtt}}$ represents the connection setup time associated with the TCP-TLS handshake and MQTT `CONNECT`. The expression $S/B + D_{\text{prop}}$ represents the total transmission (S/B) and propagation (D_{prop}) delays from the MQTT publisher to the MQTT broker, where S and B denote the payload size and link bandwidth, respectively. L_{proc} represents the processing delay at the MQTT broker. The summation represents the cumulative latency in transferring N messages.

• **MQTT over QUIC**: QUIC eliminates the full TLS handshake by integrating cryptographic negotiation into its transport handshake, thereby reducing session setup latency. Moreover, QUIC supports session resumption, where previously established cryptographic state can be reused to enable 0-RTT or 1-RTT connection establishment. This further decreases connection setup time for subsequent transmissions when the client and server share valid session tickets. In this work, we consider 1-RTT connection establishment. Furthermore, we assume periodic data transfer rather than continuous streaming, as it represents the dominant communication pattern in smart grid applications such as smart metering, energy monitoring, and demand response. Mathematically, the latency is calculated as follows:

$$L^{\text{np}}_{\text{quic}} = \sum_{i=0}^{N} \Big(\overbrace{T_{\text{quic}} + T_{\text{mqtt}}}^{\text{connection setup}} + \overbrace{S/B + D_{\text{prop}}}^{\text{transmission-propagation}} + \overbrace{L_{\text{proc}}}^{\text{processing}} \Big),$$

where T_{quic} represents the latency associated with the QUIC handshake, including possible reductions due to session resumption.

A. Measurement Procedure and Settings

- For each payload size (1 kB, 64 kB, 128 kB, 256 kB, 512 kB), we run 50 iterations of `connect-publish-disconnect` cycles for non-persistent experiments and record the elapsed time per iteration at the publisher, corroborated with the subscriber timestamps when needed. Similarly, for persistent connection, average elapsed time for each message transfer for each session with connection establishment is taken into consideration.
- Clocks on the Raspberry Pi (publisher) and UPF (broker) and the subscriber machine are synchronized using NTP (UTC) to ensure accurate end-to-end latency measurement.
- CPU and memory usage are sampled using `pidstat` on each host during experiments. Protocol-level traces are captured using `Wireshark` for validation and debugging.
- Packet captures and endpoint logs are used to compute per-iteration latency, standard deviations, and to perform statistical significance tests where applicable.

Henceforth, we use `QUIC` and `TCP-TLS` to represent the MQTT communication with QUIC and TCP-TLS, respectively.

B. Results and Discussion

The message size varies from a few bytes to tens of kilobytes in smart grid communications [11]. Consequently, we evaluate MQTT performance using payloads of varying sizes, where larger messages are fragmented into multiple segments by the transport layer to maintain protocol-level efficiency. The following subsections present the latency and resource utilization results obtained from the experiment conducted on the private 5G testbed.

1) Non-Persistent Session: Figure 4 represents the latency performance of MQTT over QUIC and MQTT over TCP-TLS obtained for non-persistent connection. QUIC improves average latency by approximately by approximately 60% on average relative to TCP-TLS. This gain is primarily due to QUIC's faster connection establishment using 0-RTT/1-RTT handshakes, whereas TCP-TLS incurs a full TCP three-way handshake followed by a TLS handshake for each transmission. Latency increases with payload size due to fragmentation into multiple segments and the corresponding increase in transmission and processing overhead. Despite this expected behaviour, QUIC maintains a clear and consistent performance advantage across all payload sizes, making it highly suitable for periodic smart grid message workloads operating over private 5G networks where sessions are frequently re-established.

2) Persistent Session: In a persistent session, `QUIC` and `TCP-TLS` yield equivalent performances in terms of average latency, as shown in Figure 5. This is because the connection is initially established for both `QUIC` and `TCP-TLS`, and data

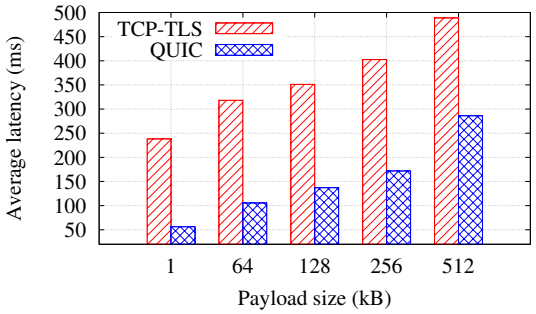


Fig. 4: Average latency in non-persistent connection with varying payload size

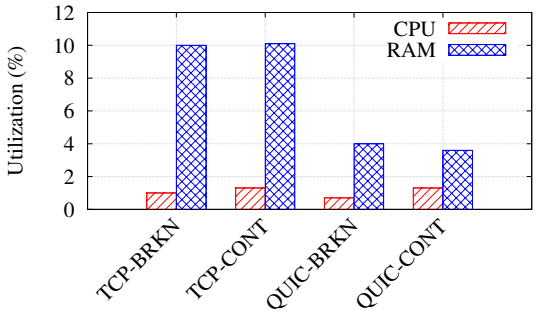


Fig. 6: CPU and RAM utilization

transfer happens continuously without closing the session for each message transfer, unlike the non-persistent connection. Additionally, the latency in TCP-TLS is reduced further due to the kernel-level optimization of the TCP protocol. Despite of this, QUIC yields competitive performance to the TCP-TLS.

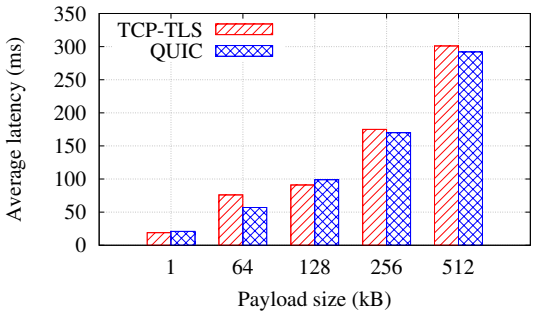


Fig. 5: Average latency in persistent connection with varying payload size

3) CPU and RAM Utilization: We measure the CPU and RAM utilization at the MQTT publisher. The CPU and RAM utilization is measured based on the CPU and RAM resources used for specific activities, such as connection establishment and data transfer using the different approaches, as mentioned earlier. Therefore, the measurement does not take into account the CPU and RAM utilization for background tasks. Figure 6 presents the CPU and RAM utilization with different connection settings – TCP-BRKN, TCP-CONT, QUIC-BRKN, and QUIC-CONT – to represent TCP-TLS with non-persistent, TCP-TLS with persistent, QUIC with non-persistent, and QUIC with persistent, respectively. The interpretation of the results on CPU and RAM utilization is discussed below.

- **CPU Utilization**: As presented in Figure 6, QUIC and TCP-TLS provide equivalent CPU utilization, which is below 15%. This low utilization makes the approach suitable for resource-constrained smart grid applications using a private 5G networks. In QUIC, connection setup, packet-level retransmissions, congestion control and encryption with TLS are handled at the user-space using the NNG QUIC library, which involves additional CPU utilization due to a large number of context switching. Furthermore, the encryption of every packet in QUIC is tightly integrated with packet-level behavior (e.g., packet numbers affect encryption). In contrast, TCP uses kernel-level optimization for connection establishment, and encryption is done only after the TCP packetization. Therefore, the CPU workload using TCP-TLS is segregated. Despite these facts, QUIC still provide lower CPU utilization than that of TCP-TLS. Furthermore, a higher CPU utilization for both QUIC and TCP-TLS is observed in persistent connections due to the resource reservation for maintaining the established session compared to the non-persistent connection.
- **RAM Utilization**: As shown in Figure 6, QUIC outperforms TCP-TLS in terms of RAM utilization for both persistent and non-persistent connections. In particular, QUIC reduces 60% RAM utilization compared to TCP-TLS. This is because QUIC, implemented at the application layer using modern and efficient packet and stream handling approaches, keeps memory usage minimal. In contrast, TCP-TLS yields a higher RAM utilization due to socket buffers, memory requirements for kernel processing, and segregated TLS encryption.

In summary, it is evident that MQTT over QUIC is beneficial for ensuring fast, reliable and secure smart grid communications. Furthermore, low CPU utilization and low memory footprint of QUIC makes it suitable for resource-constrained devices in smart grid applications.

V. Conclusion

This work investigated the performance of MQTT communication over QUIC and TCP-TLS in a private 5G testbed. The testbed includes Raspberry-Pi enabled with multiple sensors for smart grid applications, 5G radio access and core network, and device-level variability to provide an insight into practical performance. QUIC consistently demonstrated lower latency than TCP-TLS in non-persistent messaging, confirming that its lightweight connection establishment offers clear advantages for periodic traffic in smart grid communications. The additional control-plane and scheduling delays inherent to private 5G network magnified the cost of TCP's multi-step connection setup. Whereas QUIC remained efficient due to its reduced handshake overhead. Persistent sessions showed comparable steady-state latency for QUIC and TCP-TLS, as expected once the connection setup cost is amortized.

Overall, our findings validate the viability of QUIC as a transport layer protocol for smart grid communications using 5G networks. Its ability to maintain low-latency performance under real-world network dynamics, combined with efficient resource usage and integrated security, positions QUIC as a strong candidate for future scalable and latency-sensitive smart grid applications. Future work will explore multi-stream MQTT over QUIC, congestion control optimization, and throughput scaling for high-volume traffic workloads.

REFERENCES

[1] N. Kaur and L. Gupta, "Enhancing IoT Security in 6G Environment With Transparent AI: Leveraging XGBoost, SHAP and LIME," in *Proc. of the IEEE NetSoft*, 2024, pp. 180–184.

[2] F. Tonini, P. Lanci, D. Borsatti, W. Y. Poe, R. Trivisonno, and W. Cerroni, "End-to-End Performance Analysis for Intelligent IoT Devices in Goal-Oriented Networking," in *Proc. of the IEEE NetSoft*, 2025, pp. 1–6.

[3] M. Scharf and S. Kiesel, "NXG03-5: Head-of-line Blocking in TCP and SCTP: Analysis and Measurements," in *Proc. of the IEEE Globecom*, 2006, pp. 1–5.

[4] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proc. of the Conf. of the ACM Special Interest Group on Data Communication*, New York, USA, Aug. 2017, pp. 183–196.

[5] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Request for Comments RFC 9000, May 2021.

[6] P. Kumar and B. Dezfouli, "Implementation and analysis of QUIC for MQTT," *Computer Networks*, vol. 150, pp. 28–45, 2019.

[7] F. Fernández, M. Zverev, P. Garrido, J. R. Juárez, J. Bilbao, and R. Agüero, "And QUIC meets IoT: Performance assessment of MQTT over QUIC," in *Proc. of the WiMob*, Oct. 2020, pp. 1–6.

[8] S. Jeddou, F. Fernández, L. Diez, A. Baina, N. Abdallah, and R. Agüero, "Delay and Energy Consumption of MQTT over QUIC: An Empirical Characterization Using Commercial-Off-The-Shelf Devices," *Sensors*, vol. 22, no. 10, p. 3694, 2022.

[9] "NNG: Lightweight Messaging Library," Accessed on Feb. 2025. [Online]. Available: https://nng.nanomsg.org/

[10] "The quic-go Protocol Suite," Accessed on Feb. 2025. [Online]. Available: https://quic-go.net/docs/

[11] J. Momoh, *Smart Grid: Fundamentals of Design and Analysis*. Wiley, 2012.

[12] Z. Fan, P. Kulkarni, S. Gormus, C. Efthymiou, G. Kalogridis, M. Sooriyabandara, Z. Zhu, S. Lambotharan, and W. H. Chin, "Smart Grid Communications: Overview of Research Challenges, Solutions, and Standardization Activities," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 21–38, 2013.

[13] M. Kashyap, V. Sharma, and N. Gupta, "Taking MQTT and NodeMcu to IOT: Communication in Internet of Things," *Procedia Computer Science*, vol. 132, pp. 1611–1618, 2018.

[14] S. Quincozes, T. Emilio, and J. Kazienko, "MQTT Protocol: Fundamentals, Tools and Future Directions," *IEEE Latin America Transactions*, vol. 17, no. 09, pp. 1439–1448, 2019.

[15] "Paho-mqtt: MQTT version 5.0/3.1.1 client class," Accessed on Feb. 2025. [Online]. Available: http://eclipse.org/paho

[16] E. T. Inc, "EMQX: The Unified MQTT Platform for AI and IoT Data Streaming." [Online]. Available: https://www.emqx.com/en