ICT Express 11 (2025) 264–269

Contents lists available at ScienceDirect

ICT Express

journal homepage: www.elsevier.com/locate/icte

Enabling low-latency digital twins for large-scale UAV networks using MQTT-based communication framework

Dohyun An[a], Hyeontae Joo[b], Hwangnam Kim[b,*]

[a] Department of Smart Convergence, Korea University, Seoul 02841, South Korea
[b] School of Electrical Engineering, Korea University, Seoul 02841, South Korea

ARTICLE INFO

Keywords:
AoI
CMQC
Digital twin
Large UAV
Low latency
MQTT
Socket communications

ABSTRACT

In recent developments within large-scale Unmanned Aerial Vehicle (UAV) networking, ensuring the accuracy and timeliness of real-time status updates has become increasingly critical. This paper proposes a new Centralized Message Queuing Telemetry Transport (MQTT)-based Communication framework(CMQC) specifically designed to improve real-time data transmission within a simulated large-scale UAV network. The primary objective of this study is to demonstrate the superior performance of the proposed framework in comparison to traditional communication methods such as MQTT, Transmission Control Protocol (TCP) socket-based communication, and Constrained Application Protocol (CoAP). Key performance metrics such as inter-process time interval, total completion time, jitter, and Age of Information (AoI) are evaluated to substantiate these improvements. Experimental results confirm that the MQTT-based framework significantly enhances real-time update capabilities, reduces latency, and improves overall network efficiency. These findings suggest that the proposed framework is a viable solution for enabling low-latency digital twins in large-scale UAV networks.

1. Introduction

The rapid expansion of large-scale Unmanned Aerial Vehicle (UAV) networks across military, public, and private sectors has created a pressing demand for real-time, low-latency communication systems [1]. Digital twins, which represent virtual replicas of real-world environments, are becoming increasingly important for managing the high mobility and dynamic conditions of UAV networks [2,3]. However, achieving accurate and timely data transmission under resource constraints remains a significant challenge [4].

This paper introduces a novel Centralized Message Queuing Telemetry Transport (MQTT)-based Communication Framework (CMQC) designed to enhance the real-time update capabilities of UAV network digital twins [5,6]. The primary objective of this study is to evaluate the CMQC framework's performance against traditional communication methods such as MQTT, Transmission Control Protocol (TCP) socket-based communication, and Constrained Application Protocol (CoAP). Key performance metrics include inter-process time interval, total data processing time, and jitter, providing a comprehensive understanding of the framework's effectiveness in large-scale UAV networks.

In addition, we assess the freshness of transmitted data using the Age of Information (AoI) metric and evaluate the accuracy of the digital twin using the Fréchet distance, which quantifies trajectory alignment between the twin and its physical counterpart. This comprehensive evaluation highlights the CMQC framework's significant improvements in efficiency, reliability, and accuracy for real-time UAV operations.

The remainder of this paper is organized as follows. Section 2 reviews related work and compares existing communication protocols. Section 3 outlines the methodology, including system modeling, data collection, and performance evaluation. Section 4 presents the results, followed by conclusions and future work in Section 5.

2. Related work

The increasing demand for real-time communication in UAV networks has led to the development of several protocols tailored to meet the unique requirements of these systems. One of the most commonly used protocols is Transmission Control Protocol (TCP), which provides reliable, connection-oriented communication. However, TCP's high overhead and latency make it unsuitable for UAV networks, where rapid updates and minimal delay are critical. Studies have shown that TCP's connection establishment process and large control message overhead significantly increase latency in high-mobility environments, leading to performance degradation in UAV networks.

To address these limitations, lightweight protocols such as MQTT and CoAP have been widely adopted, particularly in Internet of Things

* Corresponding author.
E-mail addresses: sinsiga1905@korea.ac.kr (D. An), motern800@korea.ac.kr (H. Joo), hnkim@korea.ac.kr (H. Kim).

https://doi.org/10.1016/j.icte.2025.01.001
Received 15 July 2024; Received in revised form 3 November 2024; Accepted 6 January 2025
Available online 14 January 2025

(IoT) applications. MQTT's publisher–subscriber architecture allows for asynchronous message transfers, making it resilient in environments with unstable network conditions or limited resources. It is also designed to be lightweight, minimizing data transmission overhead and ensuring efficient resource usage. However, MQTT's focus on minimizing overhead can lead to inflexibility, especially when error handling is required. For instance, MQTT only supports five error messages for server rejection, which can make debugging and handling errors challenging in large-scale systems [7,8]. On the other hand, CoAP operates over User Datagram Protocol (UDP), which is advantageous for low-power, low-bandwidth communication but lacks the reliability needed for ensuring message delivery in environments with potential packet loss, making it less ideal for large-scale UAV networks.

In addition to these, other protocols such as HTTP and WebSockets have also been explored for UAV networks. While HTTP offers a well-understood, stateless communication mechanism, it incurs high latency due to repeated connection setup and teardown. WebSockets, although providing full-duplex communication, are less optimized for low-bandwidth scenarios typical of UAV networks. Each protocol has its strengths and weaknesses depending on the specific requirements of the application and the network environment [5,8].

Our proposed CMQC framework, based on MQTT, seeks to fill the gap left by existing protocols by offering a scalable, low-latency solution specifically optimized for UAV networks. The decision to utilize MQTT is driven by its lightweight nature and its ability to maintain real-time updates even in constrained environments. Additionally, while traditional MQTT may have limitations in terms of error handling, CMQC builds on MQTT's strengths by introducing improvements in data freshness, inter-process time interval, completion time, jitter control, making it more suitable for large-scale UAV deployments.

One key improvement in the proposed CMQC framework is the use of QoS level 2 for guaranteed message delivery, ensuring that each message is delivered exactly once, which is critical in unstable network environments like large-scale UAV operations [9]. To address MQTT's limitation of supporting only five predefined error messages for server rejections, we implemented a custom error handling mechanism. This allows detailed diagnostic messages to be exchanged through specific MQTT topics, providing more flexibility in identifying and resolving communication issues compared to standard MQTT or other protocols like CoAP. These enhancements improve both reliability and ease of debugging in large-scale UAV networks.

This study compares CMQC with other common protocols such as MQTT, CoAP, and socket communication. Through comprehensive performance evaluations, we demonstrate that CMQC outperforms these protocols in key metrics such as inter-process time interval, total completion time, and jitter, while also maintaining lower Age of Information (AoI). Furthermore, to ensure the accuracy of the digital twin in UAV networks, we introduce the use of Fréchet distance to evaluate the similarity between the digital twin and the real-world UAV system. This novel approach provides an additional layer of validation for the CMQC framework in real-time applications.

By addressing the limitations of existing protocols such as MQTT and CoAP, and comparing the performance with other widely-used protocols, this work demonstrates the potential of CMQC as an advanced communication framework for UAV networks. The results of our study confirm that CMQC not only enhances real-time data communication but also ensures higher reliability and consistency in maintaining data freshness across large-scale, dynamic UAV environments.

3. Methodology

This section outlines the setup and methodological approach used to compare the performance of the CMQC framework with other communication protocols in a UAV network context. We describe the simulation tools and environments employed to create realistic network scenarios, enabling an in-depth analysis of communication strategies and their impact on performance. This setup provides a robust framework to assess real-time data handling capabilities for large-scale UAV operations.

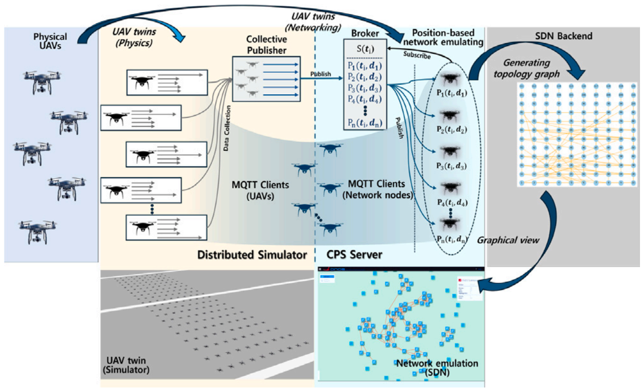


Fig. 1. Digital twin via CMQC framework.

3.1. System modeling

The system architecture of the CMQC framework is designed for real-time data transmission and low-latency communication. Fig. 1 illustrates a setup where a simulator using Webots operates UAVs, and a network emulator managed by Open Network Operating System (ONOS) handles control. ONOS, as an SDN platform, allows scalable network management. Each UAV in the simulator has a corresponding UAV in the emulator, forming a 1:1 relationship, where simulated UAVs act as MQTT clients, publishing status updates to the CPS server.

The CPS server maintains a digital twin of the UAV network, reflecting the real-time state of the UAVs, ensuring seamless synchronization between the physical UAVs and their virtual counterparts. This synchronization between the digital twin and physical UAVs allows operators to monitor the UAV in real time, facilitating real-time decision-making and mission planning based on the current UAV state.

The digital twin consists of multiple models responsible for different aspects of UAV operation, such as mobility, communication, and environmental interactions. These models work collaboratively to reflect the UAVs' behavior as closely as possible to their real-world counterparts, ensuring accuracy in real-time monitoring and predictive analysis.

The setup uses a partial-mesh network topology, allowing decentralized communication between UAVs, enhancing scalability and fault tolerance. This topology connects UAVs to the SDN backend, which converts the network topology into a graph structure, optimizing routing and control. This allows the system to dynamically manage UAV communication and ensure low-latency data transmission, critical for real-time operations.

This architecture, integrating partial-mesh topology and SDN backend, ensures the responsiveness and scalability needed for large-scale UAV operations, enabling CMQC to handle real-time data transmission efficiently while maintaining synchronization between physical UAVs and their digital twins.

In this setup, MQTT clients in the emulator subscribe to specific topics $S(t_i)$, receiving updates when the simulator-side MQTT clients publish data $P_1(t_i, d_1)$ to $P_n(t_i, d_n)$ to the same topic. This ensures timely updates between the simulator and the digital twin.

The *Collective Publisher* plays a key role by aggregating status data from multiple UAVs, periodically sending it to the MQTT broker, minimizing transmission latency and ensuring efficient data management. The broker acts as a central communication hub, supporting asynchronous messaging, even under unstable network conditions. The data from the publisher is forwarded to subscribed MQTT clients on the CPS server.

In the CPS server, a network emulator creates a digital twin of the UAV network. Subscribed MQTT clients receive real-time updates

Algorithm 1 CMQC Framework Implementation

```
Input: broker_addr, port, topic = 'drone/status'
Output: Continuously published to MQTT broker
1. Initialization:
   drone_identifiers ← []
2. Function ConnectMQTT:
   client_id = f'publish-random.randint(0, 10000)'
   client = mqtt_client.Client(client_id)
   client.connect(broker, port)
   client.loop_start()
   end
3. if drone_id is not None then
   drone_identifiers.append(drone_id)
   end
4. Connect to MQTT Broker:
   mqtt_client ← ConnectMQTT()
5. Main Loop:
   foreach drone_id in drone_identifiers do
      data ← {"drone_id": drone_id, "status": status}
   end
   Publish(mqtt_client, topic, data)
6. End
```

asynchronously, reflecting them in the digital twin, ensuring timely and accurate data representation.

By utilizing the lightweight, low-bandwidth MQTT protocol, the CMQC framework reduces transmission overhead. Its asynchronous design allows the system to scale up without performance degradation, making it well-suited for managing large UAV networks. Compared to traditional socket communication, which often suffers from higher latency and data overhead, CMQC ensures more efficient, scalable communication with high reliability.

3.2. Implementation and data collection of CMQC framework

The CMQC framework is designed around a collective publisher that orchestrates the real-time processing and dissemination of status data from a network of UAVs. Each UAV periodically sends its status data to the collective publisher, which is connected to a centralized MQTT broker. The primary role of the collective publisher is to aggregate the status data transmitted by each UAV and systematically publish this data to the MQTT broker. This approach facilitates efficient data management and minimizes the latency typically associated with large-scale data transmissions, significantly enhancing the reliability and responsiveness of the network.

The implementation process of the CMQC framework, as outlined in Algorithm 1, begins when the scenario starts, at which point all UAVs are detected. Each UAV is assigned a unique identifier and added to a list of drone identifiers. Once detection is complete, the `ConnectMQTT` function is called to assign an MQTT client to each UAV, establishing a connection with the broker and initiating the asynchronous loop to support continuous data processing. Following this, the main loop periodically generates data for each UAV. The collective publisher then asynchronously publishes the aggregated data to the MQTT broker, ensuring that updates from all UAVs are transmitted without delay and maintaining real-time data flow across the network. Following this, the main loop periodically generates data for each UAV. The collective publisher then asynchronously publishes the aggregated data to the MQTT broker, ensuring that updates from all UAVs are transmitted without delay and maintaining real-time data flow across the network.

Data collection in the CMQC system occurs as the MQTT client of the emulator subscribes to specific topics on the broker. Whenever a message is stored on the broker under these topics, it triggers reception on the client side. Callback functions are invoked upon message reception, parsing the incoming data into JSON format for ease of handling and analysis. This JSON data is crucial for both performance analysis and real-time network updates. The systematic collection and parsing process not only ensures data accuracy but also enhances the responsiveness of the network to changing conditions.

The received status data is stored for further analysis and is used to evaluate performance metrics such as inter-process time interval, total completion time, jitter, and AoI. This data collection methodology ensures that the performance of the CMQC framework is accurately assessed under realistic conditions.

3.3. AoI and mathematical modeling

AoI is a critical metric for evaluating the freshness of information in real-time systems. AoI measures the time elapsed since the last received update was generated [10]. In UAV networks, maintaining low AoI is essential for ensuring that the CPS server operates with the most recent data. At any time t, the AoI $\Delta(t)$ is defined as $\Delta(t) = t - u(t)$, where $u(t)$ is the generation time of the last received update at time t. The AoI increases linearly with time until a new update is received, at which point it drops to the time elapsed since the generation of the new update.

To model the latencies, we assume that the latencies follow a gamma distribution [10,11]. The gamma distribution is chosen due to its flexibility in modeling various types of data and its applicability to real-time systems. The probability density function of the gamma distribution is given by:

$$f_T(t;k,\theta) = \frac{t^{k-1}e^{-t/\theta}}{\theta^k \Gamma(k)}, \quad t > 0, \quad k,\theta > 0, \tag{1}$$

where k is the shape parameter, θ is the scale parameter, and $\Gamma(k)$ is the gamma function.

For gamma-distributed latencies, the expected value of the latency T is $E[T] = k\theta$.

Given that the latencies T_i follow a gamma distribution, the reception times R_i can be modeled as $R_i = S_i + T_i$, where S_i are the sending times. The AoI at reception times can be expressed as $\Delta(R_i) = R_i - S_{i-1} = (S_i + T_i) - S_{i-1}$.

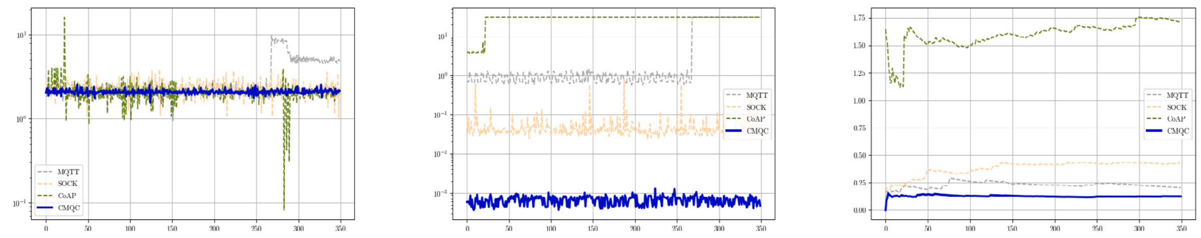
The area under the AoI curve for the interval $[R_i, R_{i+1}]$ can be computed, and by Summing these areas and normalizing by the total time T, the average AoI is:

$$\text{Average AoI} = \frac{1}{T}\sum_{i=1}^{n}\int_{R_i}^{R_{i+1}} \Delta(t)\,dt \tag{2}$$

4. Performance evaluation

In this section, we assess the performance of the CMQC against traditional socket communication, focusing on inter-process time interval, total completion time, jitter and AoI. These metrics were computed based on real-time data acquired from the network operations.

The simulation environment is constructed using Webots R2023a software on a Windows PC, equipped with an Intel(R) Core(TM) i5-14600KF processor, 32.0 GB of RAM, and an NVIDIA GeForce RTX 4060 Ti graphics card. The MQTT broker, managed by Mosquitto version 2.0.18 on a Linux laptop with an AMD Ryzen 7 7730U processor, serves as the central communication hub. Additionally, the Linux environment hosts a Python-based emulator-side MQTT client and a CPS server, which is responsible for creating digital twins of the UAVs and performing real-time network updates.



(a) Inter-Process Time Interval Comparison (s, log scale)   (b) Total Completion Time Comparison (s, log scale)   (c) Jitter(s) Comparison

Fig. 2. Performance comparison of CMQC, MQTT, Socket Communication, and CoAP.

4.1. Comparison of inter-process time interval, total completion time, and jitter

In this section, we compare the performance of CMQC, MQTT, Socket, and CoAP protocols based on three key metrics: inter-process time interval, total completion time, and jitter. These metrics were selected to evaluate the protocols' efficiency, stability, and responsiveness in large-scale UAV networks. Fig. 2 provides a visual comparison of these metrics across the different communication protocols.

The inter-process time interval $I(t)$, shown in Fig. 2(a), was defined as the time difference between the initiation of two consecutive transmissions, and is mathematically expressed as $I(t) = t_{i+1} - t_i$, where t_i is the timestamp of the ith transmission.

For CMQC, the inter-process time interval had a mean of 2.09 s with a standard deviation of 0.130 s, indicating that the transmission intervals were very close to the expected 2-second interval. In comparison, MQTT showed a mean interval of 3.02 s, with a much larger standard deviation of 1.77 s, demonstrating higher variability. The socket protocol exhibited an interval mean of 2.31 s with a standard deviation of 0.46 s, while CoAP had a similar interval mean of 2.09 s but with a higher standard deviation of 0.83 s, reflecting more inconsistency.

The total completion time T_c, illustrated in Fig. 2(b), was measured as the time taken for the protocol to process and transmit data, excluding timeout values (defined as times greater than 30 s). It is calculated as the average difference between the end time and start time for each transmission:

$$T_c = \frac{1}{n}\sum_{i=1}^{n}(t_{\text{end},i} - t_{\text{start},i}), \tag{3}$$

where $t_{\text{start},i}$ and $t_{\text{end},i}$ are the start and end times for each transmission. CMQC had the shortest completion time, averaging just 0.0007 s, highlighting its efficiency in handling real-time data. In contrast, MQTT exhibited a significantly longer average completion time of 0.8700 s, with timeouts occurring after 270 s. The socket protocol performed moderately well, with an average completion time of 0.0553 s, while CoAP had the worst performance, with an average completion time of 3.9600 s and timeouts occurring after just 35 s.

Finally, jitter, represented in Fig. 2(c), was used to quantify the variability in the latencies, which is critical for time-sensitive applications. Jitter $J(t)$ was calculated as the standard deviation of latencies:

$$J(t) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}(I_i - \bar{I})^2} \tag{4}$$

where I_i represents each latency and $\bar{I}$ is the mean latency. CMQC exhibited the lowest jitter value at 0.126 s. In comparison, MQTT recorded a jitter of 0.205 s, while the socket protocol showed a jitter of 0.434 s. CoAP had the highest jitter at 1.718 s, reflecting significant instability in its latencies.

In summary, CMQC consistently outperformed the other protocols across all metrics. As demonstrated in Fig. 2, it maintained the most consistent inter-process time intervals, the fastest completion times, and the lowest jitter, making it the most reliable protocol for large-scale UAV networks. Socket communication showed moderate stability, while MQTT had higher variability in its intervals and longer completion times. CoAP, despite being tested with fewer UAVs, showed the worst performance, marked by high jitter, long completion times, and severe limitations due to server constraints.

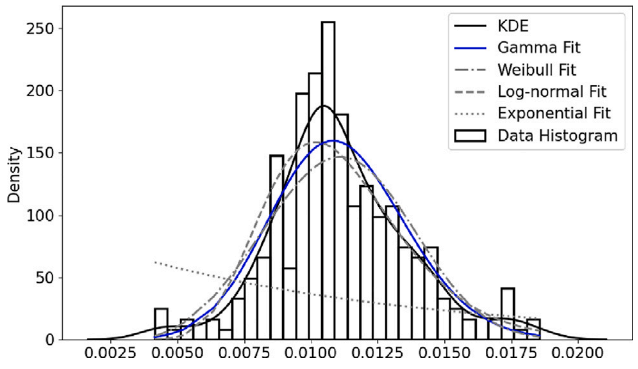


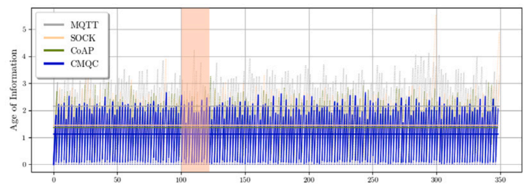
Fig. 3. Fitting of Latency Data to Various Distributions.

4.2. AoI analysis and comparison

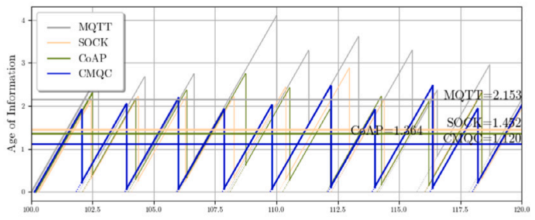
To further underscore the advantages of the CMQC framework, we performed a comparative analysis of the AoI between CMQC and traditional socket communication. As delineated in Section 3.3 , AoI serves as a critical metric for evaluating the timeliness of information in real-time systems. Our comparison is predicated on the average AoI values, which are derived from gamma-distributed latencies. The latency data set, sourced from the CMQC framework, was subjected to validation against exponential, log-normal, gamma, and Weibull distributions. The suitability of each distribution was evaluated using the Kolmogorov–Smirnov (K–S) test, which assesses how well the sample data conforms to a reference probability distribution.

The K–S test results show that the gamma distribution provides the best fit for the latency data, with a K–S statistic of 0.072 and a p-value of 0.136, outperforming the exponential, log-normal, and Weibull distributions, which had higher K–S statistics and lower p-values as illustrated in Fig. 3.

Subsequently, we utilized the gamma distribution to model the AoI for CMQC, MQTT, socket communication and CoAP. Fig. 4 illustrates the comparison of AoI values over time for CMQC and the other protocols. Fig. 4(a) shows the AoI over the entire duration of the scenario, and Fig. 4(b) zooms in on the AoI values between 100 and 120 s, providing a detailed view of CMQC's ability to maintain low AoI values during this period. The graphs clearly show that CMQC maintains lower and more consistent AoI values than MQTT, socket communication, and CoAP. The average AoI values highlight CMQC's superiority, with a significantly lower average AoI of 1.120 compared to MQTT's 2.153, socket communication's 1.452, and CoAP's 1.364. This lower AoI value indicates that CMQC provides fresher data more consistently than the other protocols, making it particularly advantageous for real-time applications where timely and accurate information is essential.



(a) AoI Over Entire Duration



(b) AoI from 100 to 120 Seconds

Fig. 4. AoI Over Time for CMQC and Socket with Gamma Distributions.

4.3. *Digital twin similarity and robustness evaluation*

By analyzing the AoI and its distribution, we can better understand the performance improvements offered by the CMQC framework in terms of maintaining the freshness of information in UAV networks. This mathematical approach provides a rigorous method for evaluating and comparing different communication protocols in terms of their impact on AoI.

We analyzed the freshness of the data received by twin UAVs based on latency from the CMQC and other groups, which updates their states based on the AoI. However, while freshness serves as a metric for data collection, demonstrating the superiority of the proposed system, it does not sufficiently indicate the sufficiency of the digital twin as an essential quality that embodies the important characteristics required for a digital twin. In this section, we present the similarity of the digital twin using the Fréchet distance [12]. The Fréchet distance allows us to measure the similarity between two curves A and B by considering the optimal reparametrization of both curves to determine the minimum leash length required for a person walking along curve A while keeping a dog on a leash walking along curve B. This is formulated as follows:

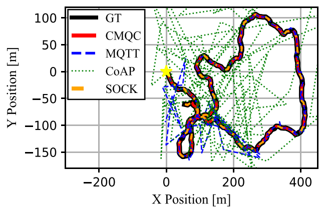
$$F(A,B) = \inf_{\alpha,\beta} \max_{t\in[0,1]} d(A(\alpha(t)), B(\beta(t))). \quad (5)$$

Meanwhile, given the latency $L = \{l_1, l_2, \dots, l_n\}$ for the ith sequence, the scaling factor k_i for the ith sequence, to compensate the latency for twin, is determined by reflecting the gradient from the previous k value, defined as follows:

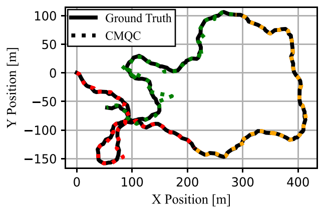
$$k_i = k_{i-1} + \left.\frac{\partial C(k_{i-1})}{\partial k}\right|_{k=k_{i-1}}, \quad (6)$$

and we perform a uniform random variation scenario with a simulated random walk at a speed of [0.5, 10] (m/s) and an angle range of $[-\frac{\pi}{4}, \frac{\pi}{4}]$ (θ). The trajectories connecting the positions of the twins are illustrated in Fig. 5(a), with the Fréchet distance-based similarity rate of 0.26% was obtained by comparing the corrected twin trajectory to the ground truth, enabling similarity performance. CMQC is very stable, and SOCK also shows similar stability. MQTT shows instability in some sections, while CoAP is unstable throughout all sections.

As the last but not least experiment, we evaluate system robustness against packet errors, we additionally simulated forced packet drops at rates of 5%, 20%, and 35%, dividing these error rates across different trajectory segments as shown in Fig. 5(b). The system maintained stability at error rates below 20% due to transmission rates with below 5% of similarity, but at 30% error, a similarity of 8.43% (> 5%) was observed.



(a) UAV and twin's trajectories



(b) UAV and twin's trajectories with error (R: 10%, Y: 20%, G: 30%)

Fig. 5. UAV random walk scenario.

5. Conclusion

In this paper, we presented the CMQC framework, a novel approach to enhancing real-time status updates in large-scale UAV networks. Our evaluation demonstrated that CMQC significantly outperforms traditional socket communication and other protocols like MQTT and CoAP in terms of inter-process time intervals, jitter, completion time, and Age of Information (AoI). These results confirm the framework's ability to maintain timely updates and ensure reliable network performance, making it highly suitable for real-time applications in dynamic and large-scale environments.

The findings indicate that CMQC offers a substantial improvement in communication efficiency and reliability, making it a promising solution for supporting digital twin applications and other UAV-related operations. Its ability to scale while maintaining performance further reinforces its potential for broader implementation across various industrial and monitoring applications.

Future work will focus on further optimizing CMQC for even larger networks, exploring more complex scenarios, and improving its robustness in challenging network conditions to ensure stable and consistent performance across diverse environments.

CRediT authorship contribution statement

Dohyun An: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Hyeontae Joo:** Validation, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Hwangnam Kim:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-2021-0-01835) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation) and the Korea Institute of Energy Technology Evaluation and Planning(KETEP) grant funded by the Korea government (MOTIE)(No. 20224B10300090).

References

[1] H. Joo, S. Lee, S. Lee, H. Kim, Optimizing time-sensitive software-defined wireless networks with reinforcement learning, IEEE Access 10 (2022) 119496–119505, http://dx.doi.org/10.1109/ACCESS.2022.3222060.

[2] Z. Qadir, K.N. Le, N. Saeed, H.S. Munawar, Towards 6G Internet of Things: Recent advances, use cases, and open challenges, ICT Express 9 (3) (2023) 296–312.

[3] Sanabria Botín, et al., Digital twin technology challenges and applications: A comprehensive review, Remote Sens. 14 (6) (2022) 1335.

[4] S. Park, K. Kim, H. Kim, Ambient virtio: IO virtualization for seamless integration and access of devices in ambient computing, IEEE Syst. J. 17 (2) (2023) 2343–2354, http://dx.doi.org/10.1109/JSYST.2022.3196624.

[5] J. Aquilina, P.A. Xuereb, E. Francalanza, J. Mallia, P. Refalo, A comparative analysis of application layer protocols within an industrial internet of things monitoring system, in: 2024 IEEE International Symposium on Measurements & Networking (M & N), 2024, pp. 1–6, http://dx.doi.org/10.1109/MN60932.2024.10615777.

[6] H. Nawaz, H.M. Ali, A.A. Laghari, UAV communication networks issues: A review, Arch. Comput. Methods Eng. 28 (2021) 1349–1369.

[7] L. Davoli, H.H.M. Ramzan, L. Belli, G. Ferrari, Coap-based digital twin modelling of heterogeneous IoT scenarios, J. Internet Things 12 (1) (2024) 15–28.

[8] B. Mishra, A. Kertesz, The use of MQTT in M2M and IoT systems: A survey, IEEE Access 8 (2020) 201071–201086.

[9] F. Hmissi, S. Ouni, SDN-DMQTT: SDN-based platform for re-configurable MQTT distributed brokers architecture, in: A. Zaslavsky, Z. Ning, V. Kalogeraki, D. Georgakopoulos, P.K. Chrysanthis (Eds.), Mobile and Ubiquitous Systems: Computing, Networking and Services, Springer Nature, Cham, Switzerland, 2024, pp. 393–411.

[10] R.D. Yates, Y. Sun, D.R. Brown, S.K. Kaul, E. Modiano, S. Ulukus, Age of information: An introduction and survey, IEEE J. Sel. Areas Commun. 39 (5) (2021) 1183–1210, http://dx.doi.org/10.1109/JSAC.2021.3065072.

[11] E. Najm, R. Nasser, Age of information: The Gamma awakening, in: 2016 IEEE International Symposium on Information Theory, ISIT, 2016, pp. 2574–2578.

[12] P. Muñoz, M. Wimmer, J. Troya, A. Vallecillo, Using trace alignments for measuring the similarity between a physical and its digital twin, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 2022, pp. 503–510.