A Combined MQTT 5.0 and DTN Architecture for Mobile Edge IoT Nodes

Vassilis Demiroglou†, Pietro Manzoni‡, Vassilis Tsaoussidis†
† *Department of Electrical and Computer Engineering, Democritus University of Thrace, Greece*
{*vdemirog, vtsaousi*}*@ee.duth.gr*
‡ *Universitat Politècnica de València (UPV), Valencia, Spain*
pmanzoni@disca.upv.es

* preprint - paper accepted to IEEE WFIoT 2023

***Abstract*—Message Queuing Telemetry Transport (MQTT) is an OASIS standard publish/subscribe protocol widely used for Internet of Things (IoT) deployments since it provides efficient many-to-many communications. IoT deployments may include mobile devices leading to unstable and intermittent wireless connections. MQTT relies on TCP and hence faces difficulties with intermittent connectivity. In support of this challenging task, Delay/Disruption Tolerant Networking (DTN) architecture handles intermittent IoT communications; however, it is incompatible with current IoT standards.**

In this work, we combine the recently released MQTT version 5.0 and DTN into an integrated architecture for edge IoT nodes to allow interoperability between continuously-connected MQTT and partially-connected DTN devices while preserving MQTT and DTN operations intact. The proof-of-concept results of our prototype implementation revealed that our solutions allow MQTT and DTN devices to stay interconnected under various disruption patterns.

***Index Terms*—MQTT, DTN, IoT, Edge Architecture**

I. Introduction

As the number of Internet of Things (IoT) devices increases, IoT technology rapidly evolves and becomes increasingly ubiquitous. Modern IoT deployments encompass various device types, including stationary sensors, battery-powered mobile nodes, and high-end edge nodes. This diverse set of devices extends coverage, enhances resilience, and provides additional computing resources. For instance, in air quality monitoring applications, stationary sensors are used for long-term observations at strategic locations, mobile devices such as drones are deployed to capture temporary measurements in inaccessible areas and achieve broader coverage, and edge stations are utilized to aggregate, store, and process the collected data locally, see [1].

Communication-wise, conventional IoT protocols should be adapted to those advances and interoperate with all types of IoT devices. Message Queuing Telemetry Transport (MQTT) is practically becoming the de-facto standard IoT protocol, featuring a publish/subscribe communication scheme and a lightweight design. However, MQTT relies on TCP for data transmission. Thus, it cannot operate efficiently upon disruptions or prolonged delays [2]. The recently released MQTT version 5.0 brings novel features and is intensively investigated for performance and flexibility improvements [3], [4].

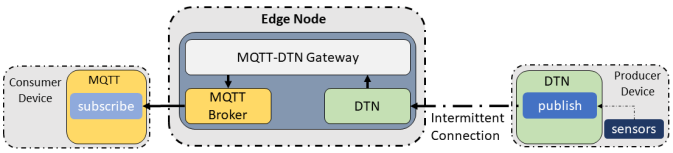
Delay/Disruption Tolerant Networking (DTN) [5] is a store-carry-and-forward architecture explicitly designed for intermittent and delayed communications. DTN enhances the performance and reliability of partially-connected IoT networks [6] but is not interoperable with other IoT standards. Therefore, combining DTN with conventional IoT protocols calls for adjustments [7], [8].

In this work, we fill these gaps by introducing a combined MQTT 5.0 and DTN architecture for edge IoT nodes that allows interoperability between continuously-connected MQTT and partially-connected DTN devices. This is accomplished through an MQTT-DTN gateway design, which is the fundamental element of our solution. The MQTT-DTN gateway operations are supported by an inter-protocol communication scheme that extends the MQTT 5.0 user properties field and adjusts accordingly the DTN bundle payload. Since the proposed solution keeps MQTT 5.0 and DTN protocols intact, it allows backward compatibility with MQTT 5.0 and DTN nodes that do not adopt our scheme.

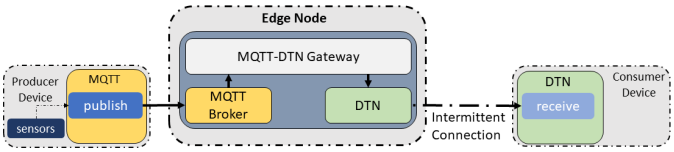
This work's contribution lies precisely in the design of a combined MQTT 5.0 and DTN architecture for edge IoT nodes. It enables the interconnection of continuously and partially connected IoT devices while keeping MQTT 5.0 and DTN operations intact. We provide a proof-of-concept evaluation of a prototype implementation to demonstrate the architecture's communication dynamics.

The results revealed that our architecture could seamlessly operate under various disruption periods, allowing MQTT and DTN devices to stay interconnected even in extreme conditions.

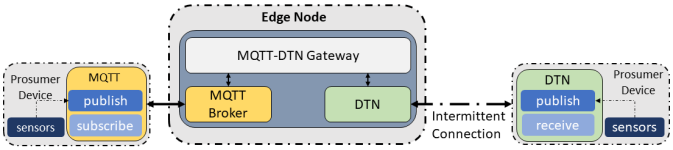
The paper is organized as follows. In Sections II and III, we discuss related works in combining MQTT and DTN protocols and provide background information of these two technologies, respectively. Section IV presents the architecture of our combined MQTT 5.0 and DTN solution for edge IoT nodes, including the MQTT-DTN gateway operations. The evaluation results are presented and analyzed in Section V. Finally, we provide conclusions and discuss future work in Section VI.



(a) Producer mobility.



(b) Consumer mobility.



(c) Producer/Consumer mobility.

Fig. 1: Motivating scenarios

II. Related Work

Recent works consider using the MQTT 5.0 user properties field to enhance the flexibility of IoT solutions in several directions. In [3], authors introduced an MQTT 5.0 geo-location extension using the user properties field. Their proposed design simplifies the integration of location-aware applications without damaging interoperability with other MQTT versions. In [4], authors introduced an Over-the-Air (OTA) updating architecture based on MQTT 5.0, focusing on end-to-end security support. As in the aforementioned MQTT 5.0 solutions, we also utilize the user properties field for keeping MQTT 5.0 protocol intact. However, our primary objective differs significantly, as we target interoperability between MQTT and DTN devices.

MQTT and DTN have been combined in various research works toward improving the reliability of conventional IoT networks. Authors in [7] and [9] introduced similar MQTT-DTN architectures for interconnection and Fog Computing support of remote MQTT deployments. These functionalities are supported through appropriate MQTT topic and DTN endpoint naming schemes. Compared to these works, our solution focuses on the interoperability between MQTT and DTN devices, follows an edge-based approach, and allows for flexible inter-protocol communications based on MQTT 5.0 extensions and DTN bundle payload adjustments.

Another group of studies combines MQTT for Sensor Networks (MQTT-SN) and DTN [8], [10]. The work in [10] involves a novel design that focuses primarily on reliability improvements of MQTT-SN via DTN; not on their interoperability. Authors in [8] propose an MQTT convergence layer adapter to DTN, allowing for cross-layer interactions. Compared to [8], our architecture handles these protocols independently but instead, facilitates the interoperability between MQTT and DTN devices through the MQTT-DTN Gateway.

III. Background

MQTT is a communication protocol that dominates the IoT research and industry area, thanks to its tiny footprint and publish/subscribe communication principles. The core MQTT functionality resides on the broker, a central node that collects messages on specific topics from MQTT publishers and forwards them to all corresponding subscribers.

MQTT was developed by IBM in 1999 and since then has been around in many variants (e.g., MQTT version 3.1 and 3.1.1). In 2019, MQTT 5.0 was introduced as a new OASIS standard. MQTT 5.0 encompasses many novel features, such as user properties, request/response support, and scalability enhancements, paving the way for more revised and flexible designs and use cases [3].

Delay/Disruption Tolerant Networking (DTN) is a networking architecture designed for challenging environments with intermittent connectivity and long delays. It employs store-and-forward communication, allowing messages to be forwarded opportunistically when a connection becomes available. DTN is particularly useful in remote areas, disaster-stricken regions, and space missions. It supports various protocols and provides message routing, fragmentation, and congestion control mechanisms. In the last decade, DTN architecture [5], [11] has been rising, and its applicability is extended besides space communications [12] to challenging terrestrial networks (such as opportunistic, *ad hoc* and sensor networks [6]). DTN architecture is based on the bundle protocol that follows a store-carry-and-forward approach to address issues of prolonged delays and intermittent connectivity [13]. This architecture provides remarkable benefits for disruptive IoT networks, and therefore, its combination with IoT standards is largely investigated [7], [8], [10].

IV. Architecture

A combination of MQTT and DTN protocols produces additional functionality that allows for more complicated IoT scenarios. Our research focuses on devising an edge architecture that accommodates different communication patterns, as illustrated in Figure 1. The first scenario, depicted in Figure 1a, involves MQTT-enabled devices retrieving sensor measurements from intermittently-connected producers. These devices act as consumers and rely on MQTT to receive data from the producers. Conversely, Figure 1b portrays the reverse scenario, where intermittently-connected consumers receive sensor measurements from MQTT-enabled devices acting as producers. This bidirectional communication pattern allows for data exchange between devices with different connectivity characteristics. Moreover, our primary objective is to design an edge architecture that can effectively handle consumer and producer mobility, as shown in Figure 1c. This scenario considers the mobility of devices that may alternate between

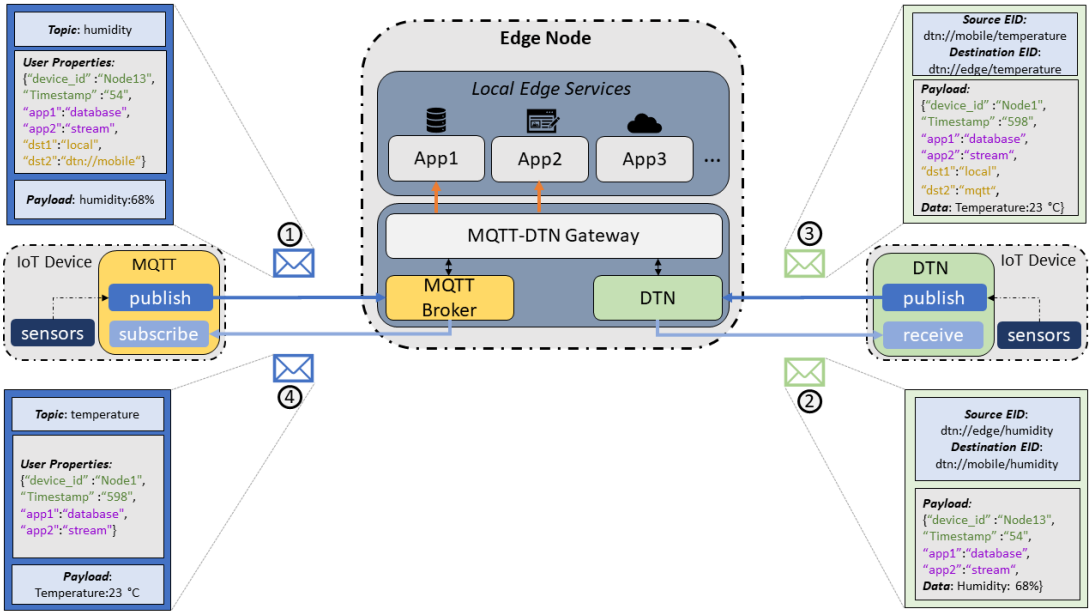


Fig. 2: Proposed architecture.

producing and consuming data, requiring seamless interoperability between MQTT and DTN protocols. By addressing these diverse communication scenarios, our research aims to enhance the versatility and robustness of IoT deployments at the edge.

The overall architecture is illustrated in Figure 2. The basic functionality of our solution is based on the edge node, which centrally collects data from MQTT and DTN devices and enables their interoperability. The edge node includes three main components: (i) the MQTT broker; (ii) the DTN daemon; and (iii) the MQTT-DTN Gateway. The edge node also supports local services, such as databases, monitoring, and streaming applications, often employed in edge IoT deployments. The MQTT broker collects IoT measurements from MQTT publishers and forwards messages to subscribers. The DTN daemon enables communication with intermittently-connected DTN devices. Thanks to these two components, the edge node can seamlessly retrieve data from continuously and intermittently connected IoT devices. In this manner, local edge services are able to aggregate IoT data generated from MQTT and DTN devices, through the MQTT-DTN gateway. The MQTT-DTN gateway component is our solution's fundamental element since it enables inter-protocol communications. In the following subsection, we elaborate on its supported operations.

A. MQTT-DTN Gateway Operations

The MQTT-DTN gateway communicates locally with the MQTT broker and the DTN daemon and processes each received MQTT message and DTN bundle individually. These messages and bundles contain additional information to enable the corresponding gateway functionality, as will be detailed in the following paragraphs. We assume that the employed edge services and the DTN endpoints are "known" by the IoT devices (e.g., through edge node advertisements or OTA firmware updates). Thus, according to the specified field(s) of each message/bundle, the MQTT-DTN gateway can: (i) translate MQTT publish messages to equivalent DTN bundles; (ii) translate DTN bundles that contain produced measurements to equivalent MQTT publish messages; (iii) forward the collected data to the specified application(s).

A main requirement of our solution is to interconnect MQTT and DTN-enabled IoT devices without sacrificing each protocol's back-compatibility. Therefore, we consider a flexible and transparent inter-protocol communication scheme that extends the MQTT 5.0 user properties field and adjusts the DTN bundle payload accordingly. This scheme is incorporated in each published message and bundle. The structure of this scheme is illustrated in Figure 2 and contains three main sections:

- *Device and measurement-specific information*: These include the *timestamp* and *device_id* fields that indicate the exact time and the device that produced the original data.
- *Application(s) selection*: The *app* field indicates the application(s) that this content should be provided (e.g., a database or a stream process).
- *Forwarding Destination(s)*: The *dst* field defines the destination(s) that this content should be forwarded.

In Figure 2, we present two relevant examples to illustrate our architecture's functionality and communication diversity in detail. In step ①, the MQTT device publishes a message to the edge node attempting to forward the contained humidity measurement: (i) to a database (*app1*) and a stream application (*app2*) running at the edge node (*dst1*); and (ii) to the specified DTN node (*dst2*). The MQTT-DTN gateway receives the MQTT message, forwards the humidity measurement to the

specified edge applications, converts it to an equivalent bundle and forwards it to *dtn://mobile/humidity*, as shown in step ②[1].

The reverse scenario is shown in step ③, where a bundle containing a temperature measurement produced from the DTN node is sent to the edge node. As in the previous example, the MQTT-DTN gateway forwards this measurement to the corresponding edge applications, creates the equivalent MQTT message, and publishes it to the MQTT broker (*dst2*). Then, the MQTT publish message is forwarded from the broker to all the MQTT devices subscribed to this topic, as shown in step ④.

V. Evaluation

To evaluate and study the dynamics of our approach, we conducted proof-of-concept experiments with our prototype implementation. To fully reflect the communication diversity of our architecture, we consider a scenario that involves a continuously-connected MQTT producer/consumer device, an intermittently-connected DTN producer/consumer device, and the edge node, as shown in Figure 2.

The following subsections present the experiment setup, the considered metrics, and the experimental results.

A. Experiment Setup

We supposed an MQTT device publishing humidity sensor measurements to the *humidity* topic and subscribed to the *temperature* topic. The DTN device sends temperature measurements to the destination Endpoint Identifier (EID) *dtn://edge/temperature* and receives humidity measurements at the *dtn://mobile/humidity* EID. In this manner, the MQTT topics and application endpoint identifiers are correctly combined. The edge node receives the produced measurements, stores the specified data in a local database (using *sqlite3*), and translates MQTT packets to Bundles and *vice-versa*.

TABLE I: Experiment Parameters

Parameter	Value
Bandwidth	1 Mbps
Propagation delay	20 ms
Disruption duration	$[1, 5, 10, 60, 300]$ sec
Connection duration	10 sec
Number of connections/disconnections	5
Number of experiment iterations	10
Publishing rate	1 msg/sec

We designed a prototype implementation of the edge node to validate the feasibility of our solution. We emulated the described scenario on a single machine (featuring an Intel Core i5200U CPU and 8 GB of RAM). All the nodes are implemented using standalone Docker containers. Specifically, the containers used for the MQTT and DTN devices utilize the Eclipse Paho MQTT Python client library and the IBR-DTN implementation, respectively. The edge node container runs our prototype edge implementation. Specifically, we developed a Docker container based on *ubuntu 14.04* image and installed the IBR-DTN [14] (version 1.0.1) and Mosquitto broker (version 1.6.3) implementations for running DTN and MQTT, respectively.

We implemented the MQTT-DTN gateway functionality in Python. We utilized the *dtnsend* and *dtnrecv* tools of IBR-DTN to send and receive DTN bundles. Also, we utilized the *paho* client library to connect to the local broker, publish MQTT 5.0 messages and subscribe to topics.

Although the edge node could support various services, for simplicity, we installed and utilized only an *sqlite3* database to store the specified data to support our architecture's basic and fundamental functionalities.

Communication of the standalone containers is accomplished through a bridge network[2]. In addition, we utilized the Linux *traffic control (tc)* tool to emulate realistic link characteristics (i.e., set the bandwidth to 1 Mbps and propagation delay to 20 msec) as well as to intermittently connect the DTN device (i.e., by alternating between *0* and *100%* packet loss).

In all the experiments, the DTN IoT device repeats a cycle of 5 disconnections and connections with the edge node. Specifically, it is connected for 10 seconds and disconnected for a specific disruption duration. To investigate the architecture performance in distinct communication conditions, we conducted a set of 5 different experiments for varying disruption duration values (i.e., $[1, 5, 10, 60, 300]$ seconds). To ensure statistically valid results, we conducted ten iterations of each experiment.

We configured both IoT devices to publish content to the edge node with a 1 message/second rate. To thoroughly investigate the dynamics of our approach, each published message/bundle has two destinations: the edge node and the other IoT device, as shown in the example of Figure 2. Thus, after the reception of a publish message/bundle, the edge node stores the payload in its local database and forwards the equivalent bundle/message to the other specified destination. Table I summarizes the main experiment parameters.

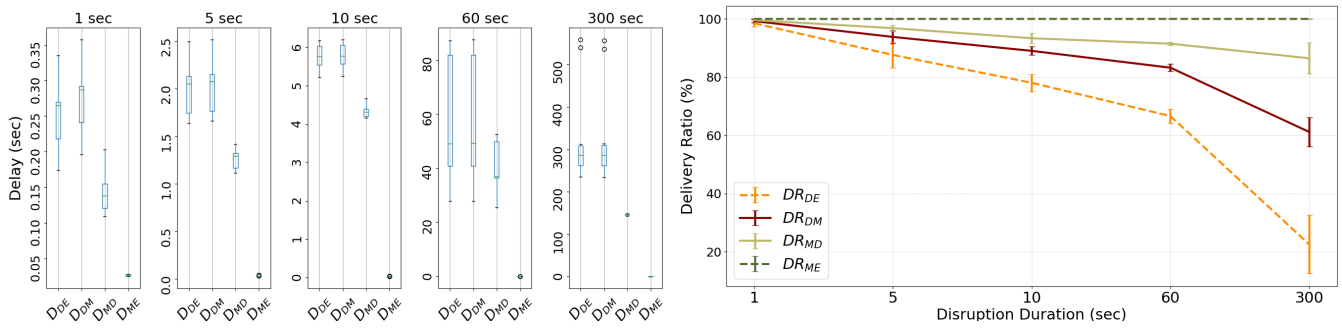
B. Metrics

We consider the following metrics:

- *Delay*: Corresponds to the time elapsed from the generation of the publish message/bundle until the reception of this measurement by the specified destination. Since each publish message/bundle has two distinct destinations (i.e., the edge node and the incompatible IoT node) it is necessary to study independently the delay of each approach. Therefore we consider the following cases: (i) D_{ME}, is the time elapsed from the generation of the publish message until the reception of this measurement by the specified edge application; (ii) D_{MD}, is the time elapsed from the generation of the publish message until the reception of this measurement by the specified DTN device; (iii) D_{DE}, is the time elapsed from the generation of the bundle that encapsulates the sensor measurement

[1]In case that the DTN IoT device supported *app1* and *app2*, the measurement could reach these services too. However, for simplicity, we consider that the MQTT and DTN IoT devices are not running any services.

[2]See: https://docs.docker.com/network/bridge/. Last accessed: 1st Jun 2023



(a) *Delay* values.

(b) *Delivery Ratio (DR)* values.

Fig. 3: Experimental results for different disruption duration.

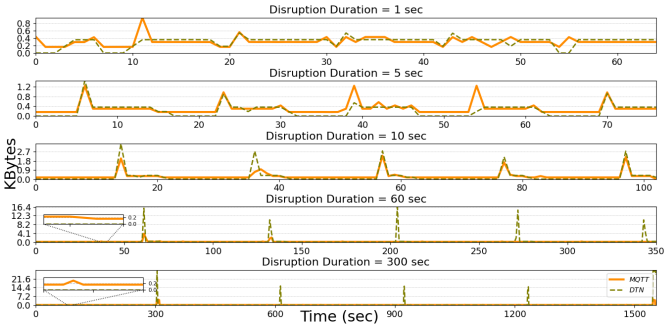


Fig. 4: Generated MQTT and DTN traffic per second.

until the reception of this measurement by the specified edge application; and (iv) D_{DM}, is the time elapsed from the generation of the bundle that encapsulates the sensor measurement until the reception of this measurement by the MQTT subscriber.

- *Delivery Ratio (DR)*: Corresponds to the ratio between the delivered and the sent messages/bundles. Like *Delay*, we consider the DR_{ME}, DR_{MD}, DR_{DE} amd DR_{DM} metrics for each particular case.
- *Communicated KBytes*: In such dynamic conditions, it is crucial to measure the generated traffic of MQTT and DTN communications and investigate their mutual influence.

Delay and Delivery Ratio metrics are measured at the receiver of each message/bundle using the timestamp field. Communicated Kbytes are calculated using *wireshark*.

C. Results

The results of our experiments are illustrated in Figures 3 and 4. In Figure 3a we present the average values of D_{DE}, D_{DM}, D_{MD} and D_{ME} for varying disruption duration values. In all cases, D_{ME} shows a stable performance (around 28 msec) irrespective of the disruption duration. This is justified because the MQTT publisher is continuously connected with the edge node.

In contrast, D_{MD} shows increased values compared to D_{ME} since each message is first published to the edge broker, stored in the local database, converted to an equivalent bundle, and finally forwarded to the intermittently-connected DTN node. Delays D_{DE} and D_{DM} are showing non-linear growth beyond 10 seconds disruptions, compared to D_{MD}. This results from the transient contacts that hinder the exchange of the increased DTN traffic. The slight difference between D_{DE} and D_{DM} is attributed to the additional delay after the bundle reception from the edge node.

In Figure 3b, we present the average values of DR_{DE}, DR_{DM}, DR_{MD} and DR_{ME} for varying disruption duration values. DR_{ME} shows increased values in each case (almost 100%) as it concerns the reliable part of our network. Although DR_{MD} is progressively reduced, it shows adequate values. This means that the communication of MQTT devices with intermittently-connected DTN devices is successfully supported.

DR_{DE} and DR_{DM} are significantly reduced as the disruption duration increases (reaching to 61% and 21% in the 300 seconds disruption, respectively). The limited contact duration and the growing number of produced bundles and messages cause this. Note that the Delivery Ratio, in our case, is measured at the end of each experiment. Thus, the decrease in Delivery Ratio does not mean that bundles have been lost, but instead, they are permanently stored at the DTN layer and could be delivered at any upcoming connection.

Figure 4 shows the Communicated Kbytes of MQTT and DTN during each experiment. In case of 1 seconds disruptions, DTN traffic is slightly smaller than the produced MQTT traffic since disconnections are short.

For longer disruptions, we observe that DTN communications occur in bursts at the start of each connection. This is because, during connection establishment, the DTN node forwards all the bundles produced through the disruption period, and the edge node forwards all the MQTT-equivalent bundles. Consequently, more significant disruptions are causing increased traffic volume exchange during contacts. This limitation causes the average D_{DE} and D_{DM} values to outreach the disruption period since some bundles are forwarded at the following contact.

VI. Conclusions and Future Work

We presented a combined MQTT 5.0 and DTN architecture for edge IoT nodes that enables interoperability between continuously-connected MQTT and partially-connected DTN devices. This architecture provides high communication flexibility, supports mobility and preserves backward compatibility with MQTT 5.0 and DTN protocols.

Our study comprehensively evaluated our MQTT-DTN combined architecture for edge IoT nodes. The experimental results provided valuable insights about the performance and behavior of the system under varying disruption durations. The stable performance of continuously-connected MQTT devices underscores their reliability, while intermittently-connected DTN devices exhibit increased delays due to transient contacts and higher traffic volumes. The high delivery ratios achieved between MQTT and DTN devices demonstrate successful communication. However, longer disruptions lead to decreased delivery ratios, revealing the challenges of intermittent connectivity. These findings contribute to a deeper understanding of the architecture's capabilities and limitations, providing valuable insights for further optimization and refinement. Overall, our proposed architecture shows promise in enabling interoperability between MQTT and DTN in edge IoT deployments.

As a future work we will focus on design extensions of our architecture to further improve the reliability and performance of intermittently-connected IoT networks, such as [13], [15]. Also, we plan to expand our architecture towards supporting distributed MQTT broker communications in remote IoT deployments. Another future direction includes the performance assessment of our architecture in large-scale and real-world environments.

Acknowledgment

References

[1] O. Cetinkaya, B. Zaghari, F. M. J. Bulot, W. Damaj, S. A. Jubb, S. Stein, A. S. Weddell, M. Mayfield, and S. Beeby, "Distributed sensing with low-cost mobile sensors toward a sustainable iot," *IEEE Internet of Things Magazine*, vol. 4, no. 3, pp. 96–102, 2021.

[2] J. E. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez, and P. Boronat, "Handling mobility in iot applications using the mqtt protocol," in *2015 Internet Technologies and Applications (ITA)*, pp. 245–250, 2015.

[3] F. Ihirwe, G. Iovino, and D. Di Ruscio, "Towards an mqtt5 geolocation extension for location-aware applications," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 100–105, 2021.

[4] H.-Y. Chien and N.-Z. Wang, "A novel mqtt 5.0-based over-the-air updating architecture facilitating stronger security," *Electronics*, vol. 11, no. 23, 2022.

[5] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, (New York, NY, USA), p. 27–34, Association for Computing Machinery, 2003.

[6] K. Loupos, A. J. Amditis, A. Tsertou, Y. Damigos, R. Gerhard, D. Rychkov, W. Wirges, Vassilis, Kalidromitis, S. Camarinopoulos, S.-A. Lenas, Tsaoussidis, A. Anastasopoulos, K. Lenz, S. Schneider, M. Hill, A. Adesiyun, and B. Frankenstein, "Skin-like sensor enabled bridge structural health monitoring system," 2016.

[7] P. Manzoni, E. Hernández-Orallo, C. T. Calafate, and J.-C. Cano, "A proposal for a publish/subscribe, disruption tolerant content island for fog computing," in *Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects*, SMARTOBJECTS '17, (New York, NY, USA), p. 47–52, Association for Computing Machinery, 2017.

[8] Y. Xu, V. Mahendran, and S. Radhakrishnan, "Internet of hybrid opportunistic things: A novel framework for interconnecting iots and dtns," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1067–1068, 2016.

[9] G. Castellano, F. Risso, and R. Loti, "Fog computing over challenged networks: A real case evaluation," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, pp. 1–7, 2018.

[10] J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni, "A disruption tolerant architecture based on mqtt for iot applications," in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 71–76, 2017.

[11] V. Cerf, S. D. Burleigh, A. Hooke, L. Gharai, E. D. Greenberg, J. R. Hughes, K. Scott, and Z. S. Bischof, "Delay-tolerant networking architecture," tech. rep., RFC 4838, April 2007.

[12] S. Burleigh, V. G. Cerf, J. Crowcroft, and V. Tsaoussidis, "Space for internet and internet for space," *Ad Hoc Networks*, vol. 23, pp. 80–86, 2014.

[13] V. Demiroglou, C.-A. Sarros, and V. Tsaoussidis, "Nod: A content retrieval scheme for intermittently-connected iot networks," *Ad Hoc Networks*, vol. 130, p. 102825, 2022.

[14] M. Doering, S. Lahde, J. Morgenroth, and L. Wolf, "Ibr-dtn: an efficient implementation for embedded systems," in *Proceedings of the Third Workshop on Challenged Networks, CHANTS 2008, San Francisco, California, USA, September 15, 2008*, pp. 117–120, Sept. 2008.

[15] V. Demiroglou, L. Mamatas, and V. Tsaoussidis, "Adaptive ndn, dtn and nod deployment in smart-city networks using sdn," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pp. 1092–1097, 2023.