EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Dimensioning payload size for fast retransmission of MQTT packets in the wake of network disconnections

Marco Domingues[1], José N. Faria[1] and David Portugal[1*]

*Correspondence:
davidbsp@isr.uc.pt

[1] Institute of Systems and Robotics, University of Coimbra, Coimbra, Portugal

Abstract

The Internet of Things (IoT) is spreading rapidly around the world, and Message Queue Telemetry Transport (MQTT) is one of the main protocols used to explore device-to-device (D2D) communication. The industry typically requires communication systems that can transmit data continuously while optimizing both bandwidth and transmission time. Due to the vast amount of data that can be lost, companies often find that even short periods of network downtime lead to significant costs. In this paper, we propose a retransmission mechanism to allow sensor nodes to relay missing data via MQTT to a local server when it reconnects after an unexpected disconnection. To assess its performance, several tests in a digital healthcare use case scenario have been designed. Since the procedure involves transferring a considerable amount of data, our main goal is to determine the maximum payload of each message to restore the missing information, while minimizing the retransmission time without information loss.

Keywords: MQTT, Retransmission mechanism, Payload size, Practical IoT applications, Performance evaluation

1 Introduction

Rapid innovations in Information and Communications Technology are leading to the wide spread of the Internet of Things (IoT). Currently, the industry needs communication systems that can reliably transmit data from a multitude of sensors, while using as little bandwidth as possible.

In most organizations, a single hour of network downtime leads to significant consequences. For instance, the loss of captured data can put people at risk or lead to financial losses; thus, one of the most important challenges in IoT is its reliability (dependability and availability).

Device-to-device communication is currently pursued through various protocols, such as the popular messaging protocol for IoT: MQTT. It was developed by [1] as a lightweight messaging transport, and it is especially suitable for connecting remote devices with minimal code and network resources. MQTT is used in many industries today, including remote monitoring [2], automotive [3], forestry [4], messaging applications [5], home automation [6] and more [7]. It is a highly developed protocol, and

among publish/subscribe communication, protocols are the one with most attempts at standardization [8].

MQTT is considered an efficient and suitable protocol designed for reliable communication between devices, particularly in the context of IoT, addressing low-power and efficient data transfer with a fast messaging model. It operates on the publish-subscribe model, where clients can either publish messages to specific "topics" or subscribe to topics to receive relevant messages. MQTT's lightweight nature makes it well-suited for resource-constrained devices and low-bandwidth networks, as it minimizes overhead by using a binary encoding scheme. This results in faster message transmission and reduced data usage, making MQTT a preferred choice for IoT devices that need to conserve power and bandwidth [9].

One of MQTT's key strengths is its reliability and scalability [10]. MQTT brokers can handle a large number of connected clients simultaneously, providing a scalable solution for IoT ecosystems. MQTT is widely adopted due to its low latency, efficient communication, and flexibility, making it a powerful protocol for enabling efficient machine-to-machine communication [7].

To prevent data loss, MQTT supports persistent sessions for broker connection [11], which allows a client to maintain its session state with a broker even if the connection is lost. This means that if a client disconnects and later reconnects, it will pick up right where it left off, with all of its subscriptions and message queues intact. This way, if the client disconnects and reconnects later, it can resume communication seamlessly. MQTT also supports retained messages [12]. When a client publishes a message with the "retained" flag set to true, the broker retains the message. Consequently, any client subscribing to the corresponding topic will receive the most recent retained message, even if no recent publications have occurred.

However, these features do not guarantee the retransmission of data when a network disconnection occurs. In such situations, all data transmitted by a publisher during the disconnection period are inevitably lost unless it retains the acquired data that could not be transmitted. Hence, it is necessary to implement additional mechanisms, such as a message queuing system or a database for message persistence.

In this paper, a retransmission functionality is proposed that allows sensor nodes to relay missing data via MQTT to servers that reconnect after an unexpected disconnection. We describe the implementation of the retransmission procedure and design several tests using a case study on digital healthcare to assess its performance. Since this procedure involves transferring considerable amounts of data, we specifically aim to dimension the optimal maximum payload size of MQTT packets for restoring the missing information in order to minimize the period of retransmission in the wake of a network disconnection.

In the next section, we provide an overview of related works that address reliability in networks with minimal data loss, and ancillary aspects. Section 3 describes the retransmission functionality proposed. Section 4 presents the case study and the experimental setup, i.e., the system used, its parameters, performance metrics and the test scenarios considered. In the following section, the experimental results obtained are analyzed, and finally, Sect. 6 presents the main conclusions drawn from this work.

2 Related work

Significant research has been conducted on achieving reliable network communications for sensor networks and IoT with retransmission abilities. In [2], an online monitoring system of multiple environmental factors for henhouses in modern chicken farms is described. The authors adopt the wireless transport Modbus protocol with a loss recovery strategy to address data packet dropout during wireless transmission. This strategy consists of making the acquisition nodes responsible for loss detection and periodically detecting the transmission flag of the device with a timeout. In case a data frame is lost during transmission, it is retransmitted back by the acquisition nodes.

Similarly, in [13], a Compressed Sensing with Dynamic Retransmission (CSDR) algorithm is proposed to guarantee the retransmission of lost data, high network lifetime and high energy utilization. The CSDR algorithm dynamically determines the maximum packet loss retransmission times of different nodes according to their residual energies. The authors verify the approach using simulated MATLAB experiments with a Wireless Sensor Network (WSN) environment and several randomly deployed nodes.

A lightweight and energy-efficient protocol called AJIA (Adaptive Joint protocol based on Implicit ACK) for packet loss recovery and route quality evaluation are described in [14]. It allows resource-constrained nodes to achieve reliable data transmission by using an overhearing feature as an implicit ACK mechanism. AJIA also provides adaptive selection of the routing path based on the connection quality. When a packet loss is detected, retransmission is carried out on the most reliable link between the node that sent the (lost) packet and its one-hop neighbors.

Two IP-based communication prototypes for reliable Industrial Internet of Things (IIoT) time-critical applications are presented in [15]. Time-sensitive networking (TSN) and edge computing are employed to increase the determinism of IIoT networks and reduce latency with zero-loss redundancy protocols that ensure sustainability of IIoT networks with smooth recovery in case of unplanned outages. The first alternative is based on the parallel zero-loss redundancy protocol (PRP) and the second one using the high-availability seamless zero-loss redundancy protocol (HSR). The PRP communication prototype goes a step further by providing an effective redundancy scheme against multiple connection failures.

Critical needs for reliable communication in wireless industrial networks are studied in [16]. A multipath routing algorithm designed to provide deterministic communication is introduced, which duplicates data flows onto alternate paths to combat potential link failures and exploit path diversity, reducing the need for retransmissions. This approach significantly outperforms traditional single-path retransmission-based methods and a state-of-the-art solution. The method achieves network reliability above 99% with ultralow jitter performance, making it a promising solution for IIoT applications that demand reliability and determinism. In connection with this, a secure data transmission technique for IIoT is presented in [17]. It introduces a blockchain-based dynamic secret sharing mechanism to enhance security and reliability. The technique ensures the reliable transmission of power data and features a consensus mechanism, dynamic linked storage, and decentralized data management. Experimental results show a significant improvement in transmission and packet receiving rates, by 12% and

13%, respectively. The proposed method not only enhances data security but also offers advantages in sharing management and decentralization.

Concerning Healthcare IoT applications, a delay-sensitive secure non-orthogonal multiple access (NOMA) transmission scheme is presented in [18]. It optimizes resource allocation to minimize information delay while securely transmitting medical data. The contributions of the paper include the introduction of a cooperative framework for healthcare services, an iterative algorithm for optimal resource allocation, and a theoretical analysis demonstrating performance superiority over orthogonal multiple access (OMA) schemes. Numerical results confirm the scheme's effectiveness in reducing secure information delay. Additionally, in [19], an energy-efficient dynamic packet downloading algorithm is introduced for improving reliability in in-hospital network architectures. The algorithm addresses the problem of network disconnections that can occur when access points are not operating efficiently, potentially leading to the loss of critical medical information. By dynamically allocating transmit power in access points based on buffer backlog size, channel states, and buffer stability, the proposed algorithm ensures a robust and reliable connection between a cloud service and healthcare IoT devices. This adaptive approach leads to improved energy efficiency and network lifetime. The study's results demonstrate that the algorithm successfully achieves the desired performance, emphasizing the importance of energy-efficient and reliable connectivity in healthcare networks.

Two alternative architectures for dealing with disconnections using the MQTT protocol are discussed in [20]. Contrarily to the work proposed in this paper, the authors tackle the problem using a redundancy approach, where additional layers are implemented consisting of dedicated MQTT brokers and critical message handlers so that the system can still operate if connectivity fails. The first of the two proposed architectures has a lightweight copy of the entire system in each sensor network, which allows the logic of the sensor nodes to be kept clean and network losses to be handled separately. In the second architecture, sensor nodes need to verify and resolve the losses of connectivity themselves, which means that they should be connected to both a local broker and an external broker.

In [21], a novel addition to TCP-based network stacks is introduced, with a focus on enhancing message transmission reliability in MQTT-based IoT applications. The proposed layer ensures reliable and in-order packet delivery, even in the presence of network or application failures, such as disconnects due to power loss. The design goals include achieving the same reliability level as Quality of Service (QoS) level 2 while reducing the number of exchanged messages, enabling multiple in-flight messages for improved bandwidth efficiency, and offering resiliency against failures. Performance tests demonstrate that the layer not only fulfills these goals but also leads to increased throughput and lower delivery latencies compared to existing protocols. The implementation was shown to handle challenging network conditions and application failures while preserving data integrity.

An edge-based MQTT broker cluster that stores queues of all messages failed to be forwarded to a remote broker is developed in [22]. The main goal is to deliver a low-cost, scalable and lightweight messaging solution to support communication between IoT devices in remote areas, where computational resources and network connectivity are

limited. The client IDs and their subscribed topics are used for message retransmission. These are advertised to other nodes in the cluster, so that they can update their local list of members. For each incoming subscription, the client ID and topics are additionally checked in a message recovery module. This module stores message queues of all publications failed to be forwarded to a remote broker.

A communication framework for IIoT that is based on MQTT broker bridging is presented in [23]. It facilitates dynamic interoperability and security across various production lines and industrial sites, addressing challenges related to reliability, scalability, and dependability. The solution ensures reliable and secure communications and offers advanced identity and access management for machines and users. Moreover, it discusses mechanisms to enhance the reliability of the broker chain by allowing MQTT implementations to be clustered on different virtual machines, thus offering fault tolerance to address overloads or failures. Results show a linear time complexity for the implementations and bridging modes of the extended brokers, with minimal overhead compared to standard MQTT brokers.

Liu and Al-Masri [24] focus on evaluating the performance and reliability of the MQTT protocol for IoT applications. A comparative study of MQTT's performance considering different payload sizes and security levels is presented. Results show that, for smaller payload sizes, higher security levels do not significantly impact latency. As message payload size increases, end-to-end communication delay also increases, and this delay is more pronounced when using QoS levels 0 and 2. However, QoS level 1 outperforms other levels in terms of performance and reliability across all security configurations and increasing message reliability with QoS level 2 does not introduce significant delays, making it comparable to QoS levels 0 and 1. This analysis provides valuable insights into optimizing MQTT for IoT applications. On the other hand, challenges in providing QoS in Electric IoT (EIoT) applications using the MQTT communication protocol are studied in [25]. The authors introduce an approach called Delay-Reliability-Aware MQTT QoS Level Selection (DR-MQLS) based on reinforcement learning. DR-MQLS optimizes QoS level selection to minimize the weighted sum of packet loss ratio and delay, considering the specific requirements of EIoT services. The proposed algorithm also enables intelligent QoS guarantee under incomplete information and enhances delay and reliability awareness, setting the stage for further optimizations, including bandwidth allocation.

A triple modular redundancy (TMR) scheme that guarantees message transmission is proposed in [26]. The system addresses reliable and fault-free sensor data to the cloud through MQTT for accurate predictions of sensor values in underground mines. The methodology focuses on identifying sensor network node failures through the use of redundant nodes. The TMR scheme consists of two backup nodes for each primary node to ensure error-free and accurate transmission of sensor data to the cloud. When a new value is received in the cloud, a deviation value is calculated. If the deviation is not within a specified allowable range, a payload mismatch occurs and the local server node determines whether the field node or the backup node is faulty.

In [27], the authors describe how devices can rely on a multi-protocol arrangement to guarantee communication success in IoT and propose a redundant architecture that targets Industry 4.0. MQTT and BLE are used as communication protocols in

their experimental environment. Results show that, independently of the protocol, the messages are sent to the framework seamlessly and users are not affected by how the messages are sent, greatly simplifying the management of redundancy applications. Furthermore, in [28], the importance of MQTT for reliable D2D communication is highlighted. The study presents a three-stage network architecture, utilizing MQTT as the transmission protocol to connect clients to a server (message broker), which securely receives, segregates, and transmits data. MQTT's reliability features, such as QoS and Last Will and Testament, are used to ensure data delivery. The proposed system combines MQTT and the MEAN stack for secure and efficient communication, offering a promising solution for IIoT scenarios.

Besides works focused on reliability, robustness, redundancy and data retransmission, there is also an important body of research that addresses lightweight communication protocols for IoT. The performance of three communication protocols (MQTT, CoAP: Constrained Application Protocol, and WebSockets) is analyzed in [29]. Comparative tests are performed, considering metrics such as protocol efficiency and Round Trip Time. Advantages and limitations of MQTT regarding energy consumption, security and reliability are discussed. Also, WebSockets and CoAP are shown to be less resilient to network volatility than MQTT, which makes it an appropriate choice in a scenario where a redundant system is important, as in the case of our work.

Davis et al. [30] analyze packet loss and reliability mechanisms in WSNs and propose an adaptive Retransmission Timeout (RTO) method to improve reaction time when packet losses occur. Two application protocols are analyzed: CoAP and MQTT-S [31], which would later become known as MQTT-SN [32], being an extension of MQTT to WSNs. The protocol is designed to be run on low-end and battery-operated sensor/actuator devices and operate over bandwidth-constraint WSNs such as ZigBee-based networks. Simulated results with OMNeT++ show that the mechanism proposed is able to significantly improve packet delivery ratio (PDR), while keeping it lightweight enough in terms of energy, memory and computation for sensor nodes where these resources are critical.

A similar study by [33] highlights the importance of setting an appropriate RTO to compensate for the lack of data reliability in WSNs. The authors propose a Gateway-assisted retransmission mechanism to dynamically compute the RTO for WSN devices. Once again, simulated experiments in OMNeT++ show that the mechanism leads to reduced retransmission times and larger message delivery ratios when compared to MQTT-SN, CoAP, dynamic RTO in TCP and CoCoA under two different message loss rates of 0% and 10%.

Focusing on resource consumption of MQTT brokers when subjected to stress testing using fuzzing techniques, Rodriguez and Batista [34] emphasize the importance of robust brokers, particularly in large-scale applications like smart cities. The results show that existing fuzzing frameworks have limitations in their ability to exhaustively test both CPU and memory resources. Only one framework, FUME, demonstrates promise in memory-intensive testing, peaking at 43.5% memory usage, highlighting the need for more robust fuzzing tools to evaluate brokers effectively, and addressing a critical aspect of MQTT's reliability. Finally, [4] explores the relevance of MQTT in addressing the challenges posed by the growing number of intelligent nodes in IoT and process

industries, including ensuring reliable data transmission while conserving network bandwidth. The study identifies MQTT as a valuable solution for managing the increasing network load and highlights best practices for implementing MQTT-based systems in industrial applications.

In spite of the existing literature on how to achieve a reliable and robust system with redundancy and retransmission features, we could not find any work which explicitly addresses optimizing the retransmission process in the wake of network disconnections for fast recovery of data accumulated (e.g., by sensor nodes) during the disconnection period using the MQTT protocol. In fact, most existing works focus on retransmission of data in unreliable networks assuming that there still is a connection link between server and client. Additionally, the majority of works in this area of research provide results based purely on simulated scenarios, which may not consider or overlook physical aspects of real-world network communications. As such, the three key contributions of this work are:

- Proposal of a novel MQTT-based Retransmission Mechanism for restoring sensor data accumulated in IoT applications during network disconnections.
- Demonstrate that there is an optimal payload size for MQTT packets that lead to faster transmission of recovered data on reconnection.
- Perform real-world experimental validation of the mechanism with commonly used hardware in a digital healthcare case study.

Based on these contributions, it is important to highlight that our approach is structured to align with the MQTT protocol and has been designed with a healthcare case study in mind. Therefore, it is not applicable to *connection-less* protocols such as CoAP. Nevertheless, we believe that its underlying principles can be adapted for other IoT scenarios with similar connectivity and data integrity requirements, such as industrial automation, smart home systems, and other sectors where a *connection-oriented* approach is favored. As such, it serves as a basis that can be adapted and extended to cater to the specific needs of IoT applications with common challenges, providing a robust foundation for researchers and developers in these domains.

3 MQTT-based retransmission mechanism for network disconnections

To establish a connection using the MQTT protocol, three entities are required: the publisher, which is responsible for sending the collected data; the subscriber, which receives the data; and the broker, which manages the data exchange [1].

It is important to note that natively, MQTT provides several countermeasures for failure scenarios, such as keep alive and message queue features, as well as Quality of Service (QoS) levels, which are agreements between the sender and the receiver of a message that defines delivery guarantees. Specifically, MQTT runs over the TCP protocol, offering a range of options for message delivery. These QoS levels, namely 0, 1, and 2, provide varying degrees of reliability and assurance for message transmission. There are inbuilt retransmission procedures for QoS 1 and QoS 2 for unacknowledged PUBLISH messages. In QoS level 1, a message is guaranteed to be delivered to the recipient at least once, but it may result in duplicate messages. In contrast, QoS level 2 provides

exactly once delivery semantics, ensuring that each message is delivered once and only once. However, achieving this high level of reliability in message delivery comes at a cost in terms of increased overhead and complexity. Therefore, it is essential for IoT systems to weigh the trade-offs and select the appropriate QoS level based on their specific use case and the potential impact of message duplication. Deduplication mechanisms at the client or server side may help to eliminate duplicate messages and minimize unnecessary processing. Additionally, message queues and acknowledgments play a pivotal role in ensuring that messages are processed in the intended order and that their delivery can be tracked. Implementing message expirations and time-to-live constraints can further enhance the efficiency of message handling. Carefully fine-tuning the QoS level, adopting appropriate mechanisms and considering the specific needs of the IoT application can optimize the communication strategy, maintaining a robust yet resource-efficient message delivery framework.

Nevertheless, some scenarios need explicit application handling, such as automatic reconnection, online buffering or throttling. This is the case of restoring data accumulated in client sensor nodes during network disconnections, as we address in this work, which cannot be solved using QoS delivery levels or other native MQTT features.

As such, we make use of the open source Eclipse Mosquito MQTT broker [35] for the retransmission mechanism proposed, due to its proven track record. Our clients use the Eclipse Paho MQTT Client [36]. Clients continuously collect sensor data and store it into a local database for redundancy, even when a disconnection to the broker occurs. The sensor acquisition database is based on MongoDB, providing high performance, high availability, and easy scalability to store raw data locally. In order to restore data accumulated while the link is suspended, when the client initiates the connection with the broker, it must define a Keep Alive period and a Will Message. The keep alive feature allows the client to periodically send a "ping" message to the broker to maintain the connection and detect if it becomes unresponsive, ensuring the connection remains active and reliable. The Will Message is used to notify subscribers, such as other clients or the broker in the event of an abnormal and unexpected client disconnection.

```
on receive(retransmission_req) do:
    buffer ← {}
    t_disc ← disconnection_timestamp()
    data_to_retransmit ← query_data_from_db(t_disc)
    for item in data_to_retransmit do
        buffer ← buffer + item
        if len(buffer) ≥ maximum_payload_size then
            publish_to_broker(buffer)
            buffer ← {}
        end if
    end for
    publish_to_broker(end_of_transmission_msg)
```

Algorithm 1 Algorithm used in the clients to retransmit the missing information to the broker.

The retransmission mechanism proposed is detailed in Algorithm 1 and Fig. 1, which is designed to restore data that was missed during a network disconnection. It collects

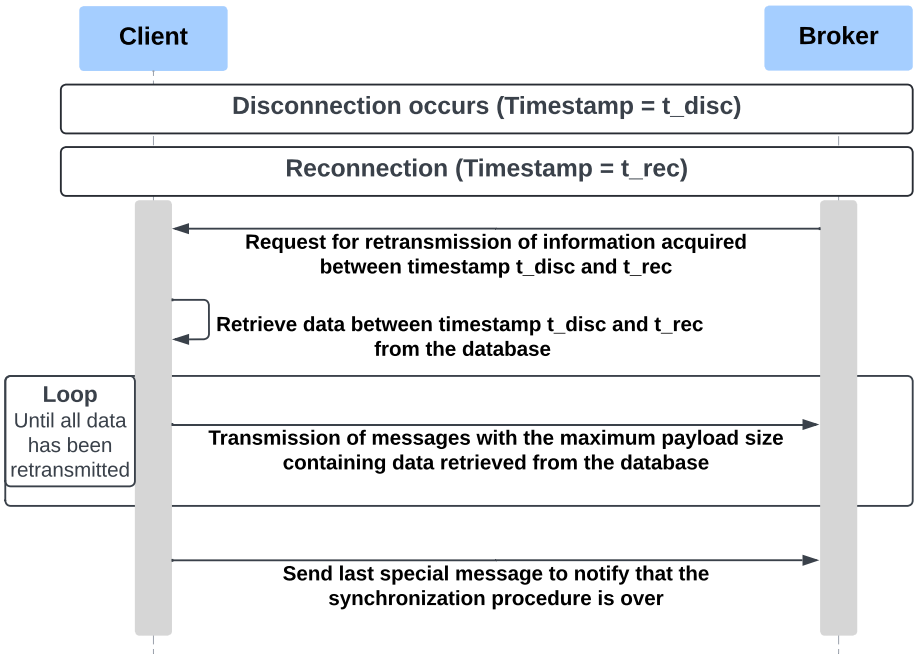

Fig. 1 Retransmission mechanism. `t_disc` corresponds to the instant when a disconnection occurs and `t_rec` to the instant when the reconnection occurs

the data, divides it into chunks of manageable size, and sends it to the MQTT broker for processing, all while ensuring that the MQTT message size limits are not exceeded.

To elaborate further on this, retransmission is triggered by the broker, which sends an initialization message through a dedicated `SYNC_REQ` topic as soon as the client reconnects (at timestamp `t_rec`). Upon receiving the retransmission request, the client initializes an empty buffer to store the data to be retransmitted, and it uses the disconnection timestamp `t_disc` to query data from the local database that needs to be retransmitted. This data corresponds to events or messages that were not transmitted due to the network disconnection. The algorithm on the client iterates through the data to be retransmitted one item at a time, accumulating multiple data items into a buffer, before transmitting them to the broker. While adding each item to the buffer, the algorithm checks whether the size of the buffer has reached the defined maximum payload size to be sent in a single MQTT message. When the buffer size meets the maximum payload size, the mechanism publishes the contents of the buffer to the MQTT broker as a message through the `SYNC_REP` topic. This ensures that the data are transmitted in manageable chunks to prevent exceeding the MQTT message size limits. After publishing the data, the buffer is reset to an empty state, allowing it to accumulate more data for the next transmission. Once all the data items have been processed and transmitted, the algorithm sends an "end of transmission" message to the broker through the `SYNC_REP_END` topic. This serves as an indicator that all missing data have been successfully retransmitted.

When data are received by the server, it saves the information in a central database, which stores data from different connected clients. We adopt PostgreSQL, which is a popular Relational Database Management System (RDBMS) database, due to the need

for a structured data relation model, especially since it is also used for the client's authentication process. PostgreSQL is an advanced, enterprise-class RDBMS, it has over 30 years of active development by the open source community, earning a strong reputation for its reliability, feature set and robustness, standing out due to its overall performance and scalability [37].

4 Experimental design

Our main aim is to determine the maximum payload size of the messages sent to restore missing data, while minimizing the retransmission time. This is done through a detailed experimental analysis.

As referred previously, we make use of a digital healthcare case study inspired in the WoW R &D project.[1] WoW proposes the development of non-intrusive wearable devices, designated as *Biostickers*, which are electronic patches equipped with sensors that acquire and wirelessly transmit patient's vital signs in real time, including body temperature, heart rate, ECG, respiration rate, as well as accelerometer and gyroscope data from an embedded Inertial Measurement Unit (IMU). *Biostickers* are low-powered and memory constrained devices that can not provide data storage capabilities, therefore they transmit sensor data to an associated "acquisition and relay" device—the *Smart box*—through Bluetooth Low Energy (BLE).

The *smart box* is embedded in each patient's bed. It encompasses a single board computer that locally acquires and stores data from the patient *biosticker*. Then, it relays the data through Wi-Fi to a central Gateway. The *Gateway* connects the *smart boxes* to the Hospital Information System, managing users and captured data, and it also maintains a list of smart boxes and sensors available. Data acquired from each patient's *biosticker* are transmitted by the corresponding *smart box* to the *gateway* via MQTT through Wi-Fi. Considering this scenario, MQTT clients run on different *smart boxes* and the MQTT broker is deployed in the central *gateway*. For more details on this architecture, the reader is referred to [38].

The retransmission mechanism proposed fits the healthcare use case chosen and similar scenarios, where *sensing nodes* are inherently limited in memory and cannot provide local data storage. Therefore, we include intermediary *acquisition nodes*, which ensure that data can be retained locally during server outages. Even though these nodes lead to the requirement of additional hardware, they provide clear advantages to overcome the constraints of *sensing nodes*. Namely, they support different wireless communication technologies for acquiring (BLE) and relaying (Wi-Fi) information, improving system robustness, and the local storage capabilities ensure that the data are not lost when the server becomes unreachable. Clearly, this design choice is especially suited for IoT scenarios where data integrity is high priority.

In addition to the retransmission mechanism described in Sect. 3, our system incorporates further improvements to the standard MQTT communication. Several features have been implemented [39], including: *i)* enhanced security by coupling TLS v1.2 encryption, X.509 V3 authentication certificates, unique UUID for each client and a

[1] WoW: Wireless biOmonitoring stickers and smart bed architecture: toWard Untethered Patients, https://inovglintt.com/financiamento/wow/.

Fig. 2 Experimental setup diagram

Table 1 Amount of data to be recovered per disconnection time

Disconnection period	30 min	1 h	2 h	6 h	12 h	24 h
Amount of data to be recovered	26.08 MB	52.50 MB	105.32 MB	316.47 MB	633.22 MB	1.23 GB

Table 2 MQTT payload sizes considered in the tests

MQTT payload size	14.84 KB	159.29 KB	445.08 KB	890.16 KB	6.52 MB	13.04 MB	26.08 MB	39.12 MB	52.16 MB
Corresponding message time	1 s	10 s	30 s	1 min	7.5 min	15 min	30 min	45 min	1 h

role-based access authorization control policy; *ii)* a pairing feature, i.e., when a *smart box* is paired with a new *biosticker*, it sends this information to the *Gateway* so that it can manage all pairings; *iii)* message format full specification based on JavaScript Object Notation (JSON) data standard to improve interoperability; *iv)* message validation and filtering at the server in compliance with the specified format before data storage and; *v)* synchronization of timestamps between *smart boxes* and the *gateway* using Chrony, an implementation of the Network Time Protocol (NTP).

The retransmission mechanism in this case study involves a *smart box* client (Raspberry Pi 4B) and the *gateway* server (Intel NUC8i7BEH) to restore the sensor data collected from a *biosticker* (nRF Dongle), when there is a Wi-Fi disconnection. Our experimental setup is illustrated in Fig. 2. Both the Raspberry Pi and the Intel NUC run Ubuntu Linux 20.04 LTS. In the experiments, a 2.4 GHz home Wi-Fi network has been used. Note that in all tests, the same data collected by the sensors used in the WoW project, i.e., IMU, ECG, Heart Rate and Body Temperature data, have been continuously provided, and the client always acquires the same 24 h of data, corresponding to 1.23 GB, as seen in Table 1.

The following configurations have been considered in the experiments:

- **Disconnection periods** of 30 min, 1 h, 2 h, 6 h, 12 h and 24 h. See Table 1 for an approximate conversion from time to the corresponding MQTT data. Disconnection periods affect the amount of data to be retransmitted. This relationship is nearly proportional, as data acquired by the client per unit of time is virtually constant.
- **Maximum payload sizes** of 14.84 KB, 159.29 KB, 445.08 KB, 890.16 KB, 6.52 MB, 13.04 MB, 26.08 MB, 39.12 MB and 52.16 MB. These have been chosen, as they correspond to specific time intervals, as shown in Table 2. Without loss of generality, the

maximum payload size is limited to 52 MB, as values above this number would lead to memory shortage errors. The maximum payload size affects how many messages the process needs for transmitting all missing data.

The proposed mechanism is tested exhaustively without considering the transmission of data being currently acquired in real time. Therefore, we subsequently look into the simultaneous transmission of real-time data together with recovered data to understand its impact on performance. Moreover, the following performance metrics have been defined:

- **Retransmission Time** (s): Time elapsed between the `retransmission request` sent by the server and its receipt of the final `end of transmission` message from the client.
- **Throughput** (Mbps): Amount of data successfully transmitted between client and server per second.

Experiments are performed by having the server requesting information to the client that correspond to the configurations defined (disconnection period vs maximum payload size). For each configuration, the procedure has been automated and repeated 100 times for statistical significance. Thus, for performance analysis, we evaluate average retransmission time and throughput.

The MQTT implementation in use adopts Nagle's algorithm [40] for TCP congestion control. This aims at improving network efficiency, minimizing the number of small packets sent over the network by delaying the transmission of these small packets through buffering and aggregation until a sufficient amount of data is available to fill the TCP window. The TCP window size, on the other hand, determines the amount of data that can be sent before requiring an acknowledgment. It is iteratively adjusted during transmissions, depending on the speed of data reception and processing at the server, whether any packets are lost in transmission, and the specifics of the underlying windowing algorithm. Both mechanisms work together to optimize the efficiency and congestion control of TCP connections to reduce overhead and congestion and naturally, they will have an impact on the results (e.g., see [41]). Hence, to ensure the exact same experimental conditions, Nagle's congestion control is enabled as the default option in Eclipse Mosquito in all tests performed, and the TCP maximum window size limit is left unmodified in the Operating System at 208 KB.

5 Results and discussion

Following the design criteria described in the previous section, a total of 4300 tests have been performed. Nearly, all configurations were deployed. However, some lower values of payload sizes have been added during the course of experiments to obtain fine-grained results.

In Figs. 3 and 4, we provide an overview of the average time required to recover the missing data in all configurations and illustrate the impact of the maximum payload size with different disconnection periods.

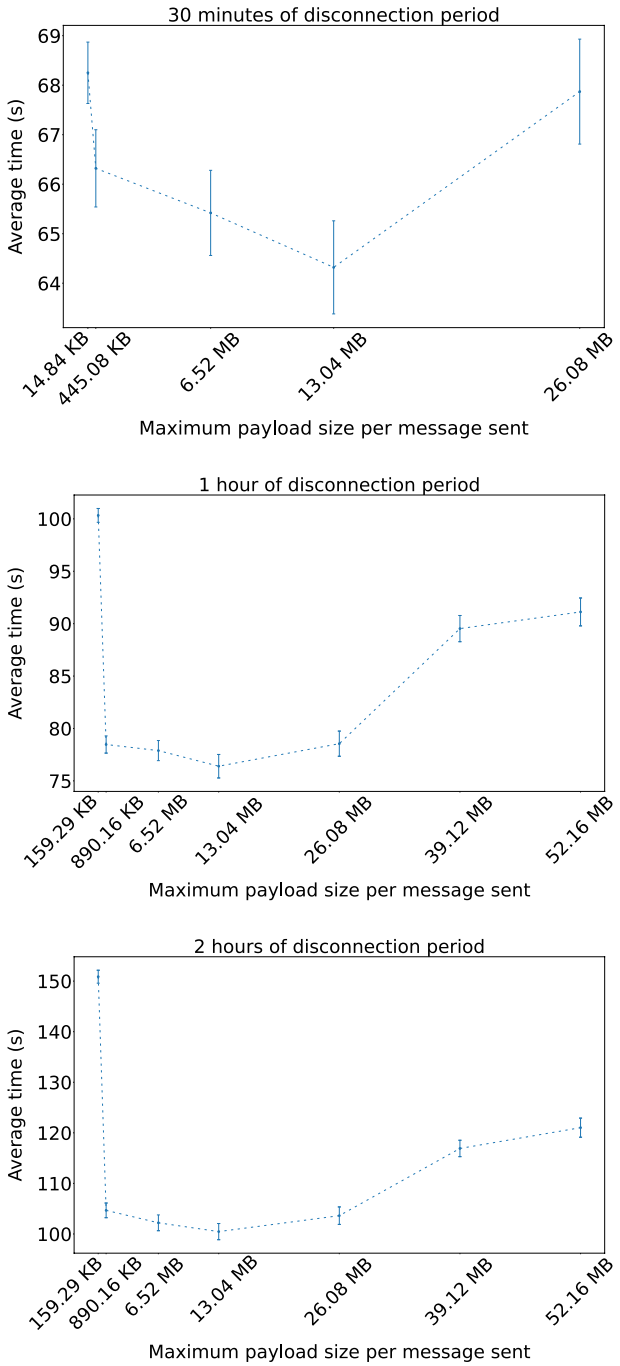

Fig. 3 Average time in seconds required to restore all missing data in disconnection periods of 30 min, 1 h and 2 h.

As expected, the longer the disconnection period is, the longer it takes until all information is retransmitted. We can also see that the time taken to restore the missing data tends to increase with the maximum payload size. However, the longest average time required is usually achieved with the smallest payload size. For instance, if we use 14.84 KB as the maximum payload size with a 30-minute disconnection period, we send

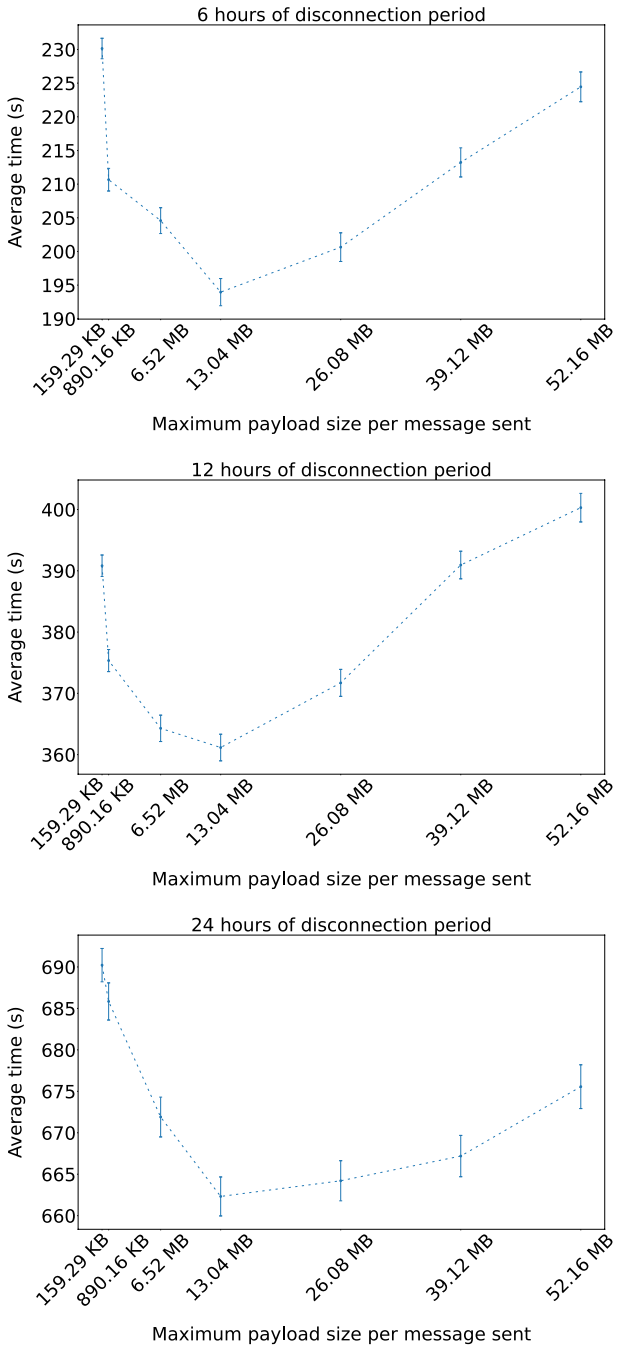

Fig. 4 Average time in seconds required to restore all missing data in disconnection periods of 6, 12 and 24 h.

about 300 messages per second, which overloads the server that runs the MQTT broker, as it is not able to receive all messages in due course and builds up a large queue, which delays the retransmission process.

Moreover, we can observe that the least time needed in all scenarios to resend the information is obtained when using 13.04 MB as maximum payload size per message, independently of the disconnection period tested. This is also confirmed by Fig. 5, which

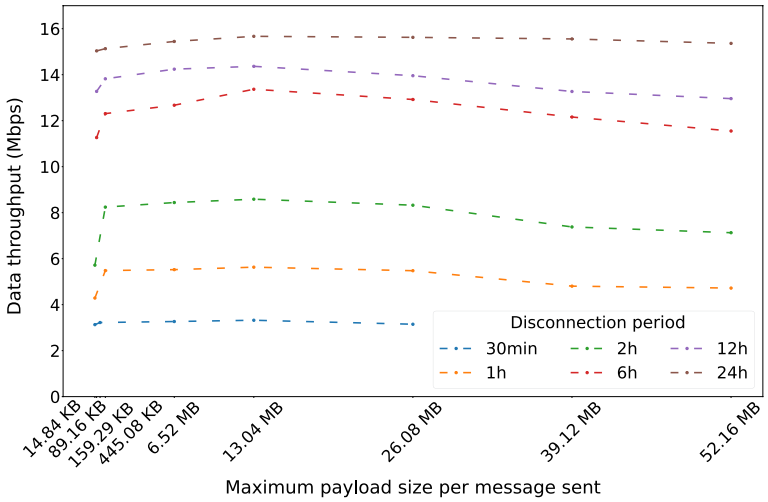

Fig. 5 Data throughput in retransmission tests (Mbps)

shows that the highest data throughput is always achieved with the same maximum payload size. This important result shows that there is an optimal payload size that leads to faster retransmission of data that are neither the minimum nor maximum limits for payload sizes of MQTT packets. This fulfills our key objective to demonstrate the existence of an optimal payload size for MQTT packets to restore the missing information in order to minimize the period of retransmission in the wake of a network disconnection.

It is worth emphasizing that the most favorable payload size of 13.04 MB found in our tests is equivalent to 15 minutes of lost data during disconnections (see Table 2). Manifestly, we can generally conclude that these results show that the optimal payload size for retransmission packets is independent of the disconnection period/volume of data to restore. Moreover, it leads to minimum retransmission time (Figs. 3 and 4) and maximum data throughput (Fig. 5) in all tests, regardless of transmission rate and amount of data to be sent by the client recovery mechanism (see Table 1). Additionally, it is also independent of the number of samples transmitted, as the performance peaks at payload sizes of 13.04 MB for 2, 4, 8, 24, 48 and 96 transmission chunks for the 30 min, 1 h, 2 h, 6 h, 12 h, 24 h disconnection periods, respectively. Still, the optimal payload for a given application is expected to depend on network configuration, link quality, transmission chain processing, software and hardware deployed. Therefore, the "sweet spot" that we identified in our case study generally may not suit all applications.

From Fig. 5, we can also see that the throughput increases with disconnection period, since contrary to what one might expect, retransmission time does not increase linearly with disconnection period. For instance, 24 h of disconnection period corresponds to approximately twice the amount of data lost when considering 12 h of disconnection. However, the retransmission time with a payload size of 52.16 MB is 675.56 seconds for the 24 h disconnection period, and 400.30 seconds for the 12 h disconnection period. As such, the rate of increase to restore the data in this situation is only of about 1.68 times.

As mentioned earlier, it is important to evaluate the impact of the simultaneous transmission of current sensor data acquired while the retransmission mechanism is in action. This is particularly important when operation of the system cannot wait for

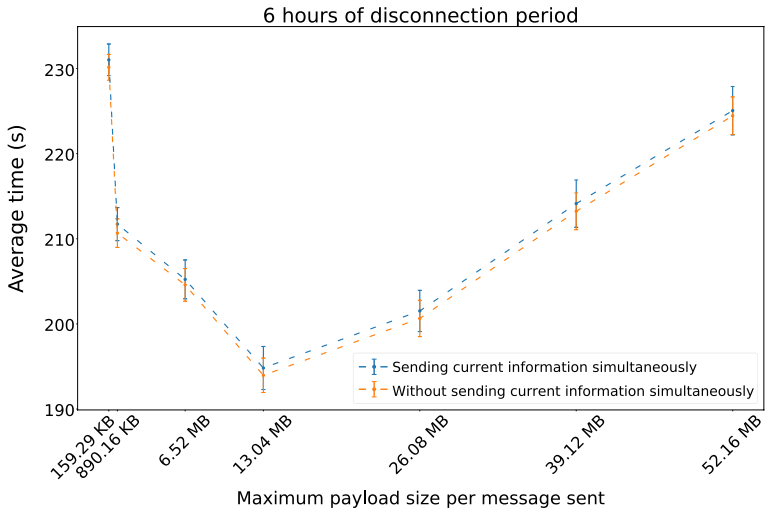

Fig. 6 Impact of also transmitting data being currently acquired

the data recovery process to finalize, such as in our case study, where critical patient data must be sent to the server for immediate access by the health professionals. From the example illustrated in Fig. 6, we can see that the simultaneous transmission of the data acquired in real time has almost no impact in the retransmission procedure for our case study. This happens because the throughput for the transmission of real-time data (175.07 kbps) is much lower compared to that of the throughput of the retransmission process (8.32 Mbps with 6 hours of disconnection period using 13.04 MB as maximum payload size per message sent) and is therefore almost negligible.

6 Conclusion

In this work, we have designed and developed an MQTT-based retransmission mechanism that allows MQTT clients to relay missing data to an associated broker, when they reconnect after an unexpected disconnection. To properly optimize the procedure, several tests that define maximum MQTT payload sizes for restoring the missing data were designed in order to study their impact on total retransmission time. The experimental assessment, in our case scenario, allowed us to dimension the payload size of the retransmission messages with a mean optimal value of 13.04 MB, which is independent of disconnection period, volume of data to restore, transmission rate and number of transmitted chunks. Furthermore, we can observe that the time required to retransmit the missing information does not increase linearly with the increase in information, and a higher throughput is typically achieved with longer disconnection periods. Additionally, we also conclude that the simultaneous transmission of the data acquired in real time has negligible impact in the retransmission feature.

In the future, it would be useful to test this functionality with additional MQTT clients. This would allow to scale up the system and study the impact of the number of clients with simultaneous retransmission processes.

Abbreviations

ACK	Acknowledgment
AJIA	Adaptive joint protocol based on Implicit ACK

BLE	Bluetooth low energy
CoAP	Constrained application protocol
CoCoA	CoAP congestion control advanced
CSDR	Compressed sensing with dynamic retransmission
D2D	Device-to-device
DR-MQLS	Delay-reliability-aware MQTT QoS level selection
ECG	Electrocardiogram
EIoT	Electrical Internet of Things
FUME	FUzzing MEssage
HSR	High-availability seamless redundancy
IIoT	Industrial Internet of Things
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internet protocol
JSON	JavaScript object notation
LTS	Long-term support
MATLAB	MATrix LABoratory
MEAN	MongoDB, Express.js, AngularJS and Node.js
MQTT	Message queue telemetry transport
MQTT-SN	MQTT for sensor networks
NOMA	Non-orthogonal multiple access
NTP	Network time protocol
OMA	Orthogonal multiple access
OMNeT++	Objective Modular Network Testbed in C++
PRP	Parallel redundancy protocol
QoS	Quality of service
RDBMS	Relational database management system
RTO	Retransmission timeout
TCP	Transmission control protocol
TLS	Transport layer security
TMR	Triple modular redundancy
TSN	Time-sensitive networking
UUID	Universal unique identifier
Wi-Fi	Wireless fidelity
WoW	Wireless biOmonitoring stickers and smart bed architecture: toWard Untethered Patients
WSN	Wireless sensor network

Acknowledgments
We would like to thank Glintt Healthcare solutions, Coimbra Hospital and University Centre (CHUC), the Soft and Printed Microeletronics (SPM) Laboratory at the Institute of Systems and Robotics (ISR) of the University of Coimbra (UC), and the Soft Machines Lab (SML) at Carnegie Mellon University for the assistance and partnership within the WoW CMU Portugal Large Scale Collaborative Project.

Author contributions
MD implemented the retransmission mechanism, developed the testbed and ran the experimental study. JF provided technical guidance and contributed toward the conceptualization and implementation of the mechanism. DP contributed toward the definition and conceptualization of the testbed. MD and DP have contributed equally to writing the manuscript, and all authors have read and approved the final manuscript.

Funding
This work was supported by the Wireless biomonitoring stickers and smart bed architecture: toward Untethered Patients (WoW) R &D project (CENTRO-01-0247-FEDER-045931), co-funded by the "Agência Nacional de Inovação", CMU Portugal program. Support for this research was also provided by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the Scientific Employment Stimulus 5th Edition, under contract 2022.05726.CEECIND.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not Applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 22 December 2022 Accepted: 12 December 2023
Published online: 02 January 2024

References

1. A.S. Clark, A Nipper, Message queue telemetry transport (mqtt) (1999). http://mqtt.org
2. H. Li, H. Wang, W. Yin, Y. Li, Y. Qian, H. Fei, Development of a remote monitoring system for henhouse environment based on iot technology. Future Internet **7**(3), 329–341 (2015)
3. A.A.O. Affia, R. Matulevičius, Securing an MQTT-based Traffic Light Perception System for Autonomous Driving, in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 255–260. IEEE (2021)
4. D.C. Mazur, R.A. Entzminger, J.A. Kay, C.A. Peterson, Analysis and overview of message queueing telemetry transport (mqtt) as applied to forest products applications, in *2021 IEEE IAS Pulp and Paper Industry Conference (PPIC)*, pages 1–7. IEEE (2021)
5. M. Köhler, D. Wörner, F. Wortmann, et al. Platforms for the internet of things–an analysis of existing solutions, in *5th Bosch Conference on Systems and Software Engineering (BoCSE)* (2014)
6. A. Flamini, L. Ciurluini, R. Loggia, A. Massaccesi, C. Moscatiello, L. Martirano, A prototype of low-cost home automation system for energy savings and living comfort. IEEE Trans. Ind. Appl. (2023)
7. Biswajeeban Mishra, Attila Kertesz, The use of mqtt in m2m and iot systems: a survey. IEEE Access **8**, 201071–201086 (2020)
8. G. Vrettos, E. Logaras, E. Kalligeros, Towards standardization of mqtt-alert-based sensor networks: Protocol structures formalization and low-end node security, in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, pages 1–4. IEEE (2018)
9. G.C. Hillar, *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd (2017)
10. M.A. Spohn, On MQTT scalability in the Internet of Things: issues, solutions, and future directions. J. Electron. Electr. Eng. **1**, 4 (2022)
11. F. Desbiens. Mqtt, in *Building Enterprise IoT Solutions with Eclipse IoT Technologies: An Open Source Approach to Edge Computing*, pages 67–101. Springer (2022)
12. D. Soni, A. Makwana, A survey on mqtt: a protocol of internet of things (iot). In *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*, volume 20, pages 173–177 (2017)
13. B. Jiang, G. Huang, F. Li, S. Zhang, Compressed sensing with dynamic retransmission algorithm in lossy wireless iot. IEEE Access **8**, 133827–133842 (2020)
14. N. Maalel, E. Natalizio, A. Bouabdallah, P. Roux, M. Kellil, Reliability for emergency applications in internet of things, in *2013 IEEE international conference on distributed computing in sensor systems*, pages 361–366. IEEE (2013)
15. K.S. Kiangala, Z. Wang, An effective communication prototype for time-critical iiot manufacturing factories using zero-loss redundancy protocols, time-sensitive networking, and edge-computing in an industry 40 environment. Processes **9**(11), 2084 (2021)
16. R.A. Koutsiamanis, G.Z. Papadopoulos, X. Fafoutis, J.M. Del Fiore, P. Thubert, N. Montavont, From best effort to deterministic packet delivery for wireless industrial iot networks. IEEE Trans Ind Inform **14**(10), 4468–4480 (2018)
17. W. Liang, M. Tang, J. Long, X. Peng, X. Jianlong, K.-C. Li, A secure fabric blockchain-based data transmission technique for industrial internet-of-things. IEEE Trans. Ind. Inform. **15**(6), 3582–3592 (2019)
18. D. Wang, Y. He, Y. Keping, G. Srivastava, L. Nie, R. Zhang, Delay-sensitive secure NOMA transmission for hierarchical HAP-LAP medical-care IoT networks. IEEE Trans. Ind. Inform. **18**(8), 5561–5572 (2021)
19. Joongheon Kim, Energy-efficient dynamic packet downloading for medical iot platforms. IEEE Trans. Ind. Inf. **11**(6), 1653–1659 (2015)
20. O. Kovalchuk, Y. Gordienko, S. Stirenko, The impact of mqtt-based sensor network architecture on delivery delay time, in *2019 IEEE 39th International Conference on Electronics and Nanotechnology (ELNANO)*, pages 838–842. IEEE (2019)
21. D. Fuchs, *A Resilient Transport Layer for Messaging Systems*. MSc. Thesis, ETH Zurich, Institute for Pervasive Computing (2007)
22. Z. Ying Thean, V. Voon Yap, P. Chiong Teh, Container-based mqtt broker cluster for edge computing, in *2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, pages 1–6. IEEE (2019)
23. Michele Amoretti, Riccardo Pecori, Yanina Protskaya, Luca Veltri, Francesco Zanichelli, A scalable and secure publish/subscribe-based framework for industrial IoT. IEEE Trans. Ind. Inf. **17**(6), 3815–3825 (2020)
24. Y. Liu, E. Al-Masri, Evaluating the reliability of mqtt with comparative analysis, in *2021 IEEE 4th International Conference on Knowledge Innovation and Invention (ICKII)*, pages 24–29. IEEE (2021)
25. H. Zhang, H. Zhang, Z. Wang, Z. Zhou, Q. Wang, X. Guangyuan, J. Yang, Z. Gan, Delay-reliability-aware protocol adaption and quality of service guarantee for message queuing telemetry transport-empowered electric internet of things. Int. J. Distrib. Sensor Netw. **18**(5), 15501329221097816 (2022)
26. A.S. Pillai, G.S. Chandraprasad, A.S. Khwaja, A. Anpalagan, A service oriented iot architecture for disaster preparedness and forecasting system. Internet Things **14**, 100076 (2021)
27. D.R.C. Silva, V.S.S. Lima, H.B.M. Alves, R.N. Cunha, E. Sisinni, P. Ferrari, Iot framework with flexible management of multi-protocol nodes for redundancy applications, in *2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4. 0 &IoT)*, pages 677–681. IEEE (2021)
28. N. Chandeliya, P. Chari, S. Karpe, D.C. Karia, Reliable machine to machine communication using mqtt protocol and mean stack, in *Innovative Data Communication Technologies and Application: ICIDCA 2019*, pages 94–100. Springer (2020)
29. S. Mijovic, E. Shehu, C. Buratti, Comparing application layer protocols for the internet of things via experimentation, in *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–5. IEEE (2016)
30. E.G. Davis, A. Calveras, I. Demirkol, Improving packet delivery performance of publish/subscribe protocols in wireless sensor networks. Sensors **13**(1), 648–680 (2013)
31. U. Hunkeler, H.L. Truong, A. Stanford-Clark, Mqtt-s-a publish/subscribe protocol for wireless sensor networks, in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pages 791–798. IEEE (2008)

32. S. Rizqika Akbar, K. Amron, H. Mulya, S. Hanifah, Message queue telemetry transport protocols implementation for wireless sensor networks communication-a performance review, in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, pages 107–112 IEEE (2017)
33. H.-L. Chang, C.-G. Wang, W. Mong-Ting, M.-H. Tsai, C.-Y. Lin, Gateway-assisted retransmission for lightweight and reliable iot communications. Sensors **16**(10), 1560 (2016)
34. L. Rodrigues, D. Batista, Resource-intensive fuzzing for mqtt brokers: State of the art, performance evaluation, and open issues. IEEE Netw. Lett. (2023)
35. R.A. Light, Mosquitto: server and client implementation of the MQTT protocol. J. Open Sour. Software **2**(13), 265 (2017)
36. Eclipse Paho, December 2022. https://www.eclipse.org/paho/
37. C. Asiminidis, G. Kokkonis, S. Kontogiannis, Database systems performance evaluation for IoT applications. Int. J. Database Manag. Syst. (IJDMS) **10**, 7 (2018)
38. F. Famá, J.N. Faria, D. Portugal, An iot-based interoperable architecture for wireless biomonitoring of patients with sensor patches. Internet Things **19**, 100547 (2022)
39. D. Portugal, J.N. Faria, M. Domingues, L. Gaspar, Integration of a smart bed infrastructure with hospital information systems using fast health interoperability resources, in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE (2023)
40. John Nagle, Congestion control in IP/TCP internetworks. ACM SIGCOMM Comput. Commun. Rev. **14**(4), 11–17 (1984)
41. E. Roques Gomez, M. Davis, The impact of TCP sliding window on the performance of IEEE 802.11 WLANs, in *2006 IET Irish Signals and Systems Conference*, pages 231–234. IET (2006)

Publisher's Note