Journal of Techniques, ISSN: 2708-8383, Vol. 8, No. 1, 2026, Pages 28-42
https://doi.org/10.51173/jt.v8i1.2790

JOURNAL OF TECHNIQUES
Journal homepage: http://journal.mtu.edu.iq

REVIEW ARTICLE - ENGINEERING (MISCELLANEOUS)

A Comprehensive Review of Quality of Service Enhancement Techniques in Message Queuing Telemetry Transport Protocols for IoT Applications

Raghad Mohammed Nayef[1], Abdulrahman Ikram Siddiq[2*], Noaman Mohammad Noaman[3]

[1]Computer Techniques Eng. Dept., Technical Engineering College, Northern Technical University, Kirkuk, Iraq

[2]Electronic and Control Techniques Eng. Dept., Technical Engineering College, Northern Technical University, Kirkuk, Iraq

[3]Mechatronics Engineering Department, College of Engineering, University of Technology Bahrain, Salmabad, Kingdom of Bahrain

[*] Corresponding author E-mail: draisiddiq@ntu.edu.iq

Article Info.

Article history:

Received
20 November 2025

Revised
03 February 2026

Accepted
25 February 2026

Published
31 March 2026

Abstract

The most popular messaging protocol in Internet of Things (IoT) networks is the Message Queuing Telemetry Transport (MQTT). It provides three Quality of Service (QoS) levels for reliable data delivery depending on application needs. Therefore, MQTT is widely adopted and supported by major IoT platforms like AWS IoT, Azure IoT Hub, IBM Watson IoT, and Google Cloud IoT. QoS enhancement techniques are needed in IoT networks because IoT devices and their applications operate in highly constrained and dynamic environments where standard communication protocols (like MQTT) alone cannot always guarantee reliability, timeliness, or efficiency. This study presents a comprehensive review of the recent and current literature on QoS enhancement techniques of MQTT-based IoT systems. These techniques are classified into five primary categories, which are: QoS enhancements at the protocol level, QoS enhancements at the network level, QoS enhancements at the broker level, cooperative subscriber approaches, and adaptive approaches based on context or machine learning. The related works belonging to these categories are explored and discussed in detail, focusing on the trade-offs of delivery reliability, protocol overhead, and energy efficiency. In addition, the study identifies the existing research gaps and limitations, and proposes future directions for advancing scalable, intelligent, and context-aware QoS management in MQTT-based IoT applications.

Publisher: Middle Technical University

Keywords: MQTT; Internet of Things (IoT); Cooperative Subscribers; Quality of Service (QoS); Publish/Subscribe; Adaptive Communication; Protocol Optimization.

1. Introduction

Internet of Things (IoT) systems are networks of interconnected devices that have a tendency to be resource-constrained and unreliable networks. To deal with this complexity, as shown in Fig.1, the publish/subscribe model has emerged as the preferred communication pattern that decouples both the data producers and consumers with a central broker. The trend supports scalability and asynchronous communications, which play a vital role in IoT environment dynamics [1, 2]. Scalability is particularly important in applications such as smart buildings [3, 4] and smart cities [5], where publish/subscribe-based patterns support energy efficiency and the coordination of large-scale systems [6].

In contrast to traditional request/response protocols, publish/subscribe enables the efficient dissemination of information to multiple subscribers with minimal overhead [7, 8]. It is also suitable to use in applications such as smart homes, industrial control, and environmental monitoring, as devices often enter and leave the network in loosely coupled interactions [9,10]. The most widely used messaging protocol in IoT is the Message Queuing Telemetry Transport (MQTT) protocol [2]. It is a lightweight protocol that is to be applied in small devices and low-bandwidth networks [11]. Its core architecture comprises three components: the publisher, which sends messages; the subscriber, which receives them; and the broker, which handles message routing. This design simplifies communication while maintaining efficiency, contributing to MQTT's widespread adoption in IoT applications [12, 13]. The protocol also provides support to the control packets required, such as CONNECT, PUBLISH, and SUBSCRIBE, to allow the clients to establish sessions, transmit messages, and create topic-based communications channels [14]. Due to its minimal overhead and simplicity, MQTT is well-suited for scenarios such as agriculture, smart healthcare, and smart cities [15, 16].

In contemporary smart-city deployment contexts, MQTT has been seamlessly incorporated into open-source platforms, thereby facilitating full-stack, end-to-end communication among urban IoT nodes [6]. The literature further underscores MQTT's suitability for lightweight applications owing to its minimal handshake protocol and efficient topic-based routing [17].

Nomenclature & Symbols			
IoT	Internet of Things	NS-3	Network Simulation Package
MQTT	Message Queuing Telemetry Transport	UDP	User Datagram Protocol
QoS	Quality of Service	COTS	Commercial-Off-The-Shelves
AWS	Amazon Web Services	MCU	Microcontroller Unit
SSL/TLS	Secure Sockets Layer / Transport Layer Security	DHT	Digital Humidity Temperature
AI	Artificial Intelligence	OTA	Over The Air
ML	Machine Learning	MQTT-SN	MQTT Sensor Network
CoAP	Constrained Application Protocol	MQTT v5	MQTT Version 5
LwM2M	Lightweight Machine-to-Machine	CPU	Central Processing Unit
TCP/IP	Transmission Control Protocol/Internet Protocol	ESP8266	Wi-Fi Microchip
LoRa	Long Range	6LoWPAN	IPv6 / Low-Power Wireless Personal Area Networks
IEEE	Institute of Electrical and Electronics Engineers	ECDSA	Elliptic Curve Digital Signature Algorithm
ACM	Association for Computing Machinery	SHA	Secure Hash Algorithm
MAC	Media Access Control	ORT	Overlay Routing Tables
EMQX	Erlang Message Queue X	PRT	Publication Routing Tables
RL	Reinforcement Learning	RTU	Remote Terminal Unit
PER	Packet Error Rate	BLE	Bluetooth Low Energy
MUP	MQTT User Properties	MLR	Message Loss Rate
RTT	Round Trip Time	5G	Fifth Generation Network
NB-IoT	Narrow Band IoT	MAB	Multi-Armed Bandit
PrioMQTT	Priority-based MQTT	XMPP	Extensible Messaging and Presence Protocol

Resilience of data and transmission efficiency is also an important feature of IoT architectures particularly as it relates to mission-critical operations-remote healthcare monitoring, industrial automation, and environmental control [18]. These environments tend to make available energy-limited devices, lose connections, or work on wireless links of low quality. This type of condition makes packet loss or latency intolerable [19, 20]. In such cases, protocols used in communication should provide data reliably and with energy efficiency. A trade-off between assurance and performance is necessary in power-sensitive devices, or real-time decision-making. Therefore, delays or loss of messages can trigger system breakdowns or unsafe behavior [4]. Therefore, applications in the smart-home domain and remote environmental monitoring systems depend heavily on MQTT's capacity to transmit temperature and security-related data with reasonable latency or loss [21]. Large-scale smart-city installations require end-to-end service assurance in MQTT-based communication, a challenge mathematically modeled by Ali and Zafar [4].

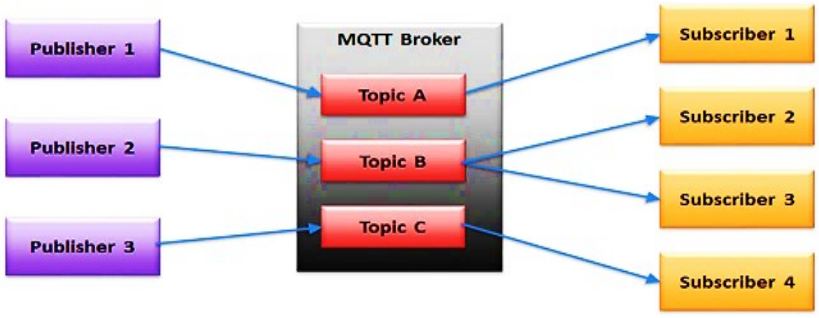

Fig. 1. Publish/Subscribe model

MQTT supports three Quality of Service (QoS) levels to accommodate different messaging reliability requirements in IoT and other lightweight communication scenarios. Each QoS level balances reliability, latency, and network overhead based on use case needs. Each level defines a unique way through which the exchange of messages between broker and client is controlled. QoS 0, commonly termed "at most once," dispatches data without acknowledgment or retry, a mode suited to non-critical communications where temporal fidelity outweighs reliability. By contrast, QoS 1, known as "at least once" mandates return confirmation, making duplicate messages possible if retransmissions become necessary. The most stringent level, QoS 2, known as "exactly once", aims to deliver messages by means of a four-way handshake that prevents duplicates, but has significantly higher additional processing and transmission overheads compared to the other two levels [13, 15]. These QoS levels provide the application developers with fine-grained control of reliability. However, the allocation of QoS levels is fixed, therefore, it does not consider dynamic network characteristics, or changing power constraints, as experienced in large-scale IoT deployments. Such brittleness is reinforced by recent empirical studies that showed the inefficiency of fixed QoS approaches in the face of changing loads and links [10, 22, 23]. Besides, existing assessments using deep learning algorithms showed that a fixed QoS assignment tends to be suboptimal, particularly in cases of unstable workload or network conditions [24]. Moreover, safety-critical IoT use-cases, such as emergency alerting, expose limitations of the standard QoS framework, as underscored by priority-enabled MQTT extensions that enhance timely and prioritized messaging [18, 25]. Though the three conventional QoS levels provide the necessary control over the delivery of messages, they do not suffice in most modern IoT implementations [26]. As an example, QoS 2, which is the only one capable of ensuring the integrity of the messages, has a significant overhead because it comprises a four-step acknowledgment scheme, which makes it less practical to use with the constrained devices that have a difficult time meeting the strict processing [27] and energy requirements [28].

Although the MQTT protocol is characterized by significant utility, the security aspects adopted by MQTT brokers are highly heterogeneous. As another example, the open-source Mosquitto broker also implements the basic security features like SSL/TLS (Secure Sockets Layer / Transport Layer Security) and client certificates; however, they are not enabled by default. In the default setting, sensitive authentication credentials are passed in unencrypted form, thus presenting an urgent vulnerability. This restriction indicates a generic vulnerability that is

characteristic of the default QoS controls in MQTT that do not have an implied end-to-end security unless supplemental security controls are employed [29]. Analogously, [30] observes that MQTT brokers frequently operate as passive routers, lacking intelligent buffering or priority-based strategies that could enhance responsiveness in dynamic settings. These limitations lower the scalability and responsiveness especially in massively deployed and real-time data processing [31]. In environments with a high density of concurrent subscribers, the default QoS mechanisms often lead to delivery failures or delays, as evidenced by reliability studies of many-subscriber MQTT networks [23]. Empirical evaluations of MQTT client libraries in industrial contexts further demonstrate that performance is highly sensitive to QoS selection and that current standards fall short in addressing real-time demands [31]. Considering MQTT's growing ubiquity in numerous IoT applications, addressing its QoS limitations has become a focal point in recent research. Multiple studies propose enhancements across various protocol layers. As an example, the work in [32] and [10] emphasizes the necessity of a strict performance measurement in the comparisons of IoT communication protocols, which points to the necessity to be precise in the methodological approach to the evaluation of the improvement at the protocol level. An extensive survey of IoT-cloud integration was presented in [33] and it emphasized the role of QoS in systems of this type.

Although several review articles [1- 3] have examined MQTT and QoS mechanisms in IoT, most existing surveys mainly focus on protocol-level performance, broker architectures, or general reliability and security aspects. In contrast, this review places particular emphasis on emerging techniques such as cooperative QoS enhancement by subscriber-side retransmission, message caching, and peer-assisted recovery. These approaches introduce a new paradigm in MQTT systems in which reliability is not solely enforced by the broker but is collaboratively supported by end devices. This application-driven and cooperation-oriented perspective distinguishes the present review from existing literature and highlights open challenges related to synchronization, security, and scalability that have not been sufficiently addressed in prior surveys.

This review systematically consolidates and evaluates the heterogeneous literature dedicated to enhancing message-oriented QoS in MQTT-based systems. The principal contribution lies in delineating a clear taxonomy that spans protocol-level modifications, intelligent broker configurations, and subscriber cooperation frameworks. The paper is also broad and deep insofar as it reviews the exemplary approaches, including the work of Fauzan [34] on the cryptographic overhead on resource-constrained devices and the work of Zunino [30] on redundancy-aware broker architectures. The taxonomy is merged with a comparative assessment that looks at reliability, latency, energy efficiency, and computational overhead, therefore, pointing out areas susceptible to trade-offs. The discussion also identifies current research gaps, including the limited deployment of adaptive QoS mechanisms in real-world testbeds and the absence of standardized support for advanced QoS features in MQTT v5.

2. MQTT Protocol Overview

MQTT is an Application Layer protocol [35] specifically designed for IoT and M2M communication. It specifies the manner in which IoT devices exchange messages, typically over TCP/IP (Transmission Control Protocol/Internet Protocol), and implements the publish/subscribe communication pattern. The protocol's control flow is relatively straightforward. A client uses a CONNECT packet to begin the process of communication with the broker, thus initiating a session. Once connected, the client may forward PUBLISH packets that publish messages to designated topics or SUBSCRIBE packets that express interest in particular topics. Furthermore, MQTT supports message retention through the RETAIN flag and provides session persistence via the clean session attribute. In order to be reliable, the PINGREQ messages are periodically transmitted, whereas the DISCONNECT packet indicates a graceful end of a session [15].

The protocol employs a finite set of control packets, as shown in Fig. 2, that collectively provide a concise and efficient communication model, rendering MQTT particularly well-suited to scenarios where bandwidth consumption and energy expenditure are critical [15, 36]. The lightweight nature of MQTT allows it to be integrated with low-power wireless modalities like Bluetooth Low Energy (BLE), thus allowing location-aware services in indoor IoT deployments [37]. Moreover, the modular structure of the protocol allows security policies to be applied at the object level, which makes the privacy-sensitive applications flexible [38]. To regulate the reliability of message delivery, MQTT provides three QoS levels. Each level prescribes distinct delivery guarantees; Fig. 3 illustrates the three MQTT QoS levels discussed as follows:

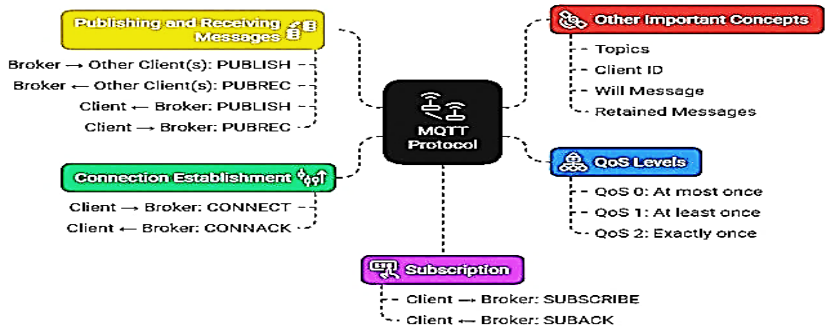

Fig. 2. MQTT protocol: Message flow and concepts

QoS 0 (at most once) does not even notify of the message. It has the lowest latency and the lowest overhead, but it does not guarantee delivery. This level is commonly used where occasional data loss can be accepted, e.g., sensor streams where it is acceptable to miss short periods of time in the data transactions. QoS 1 (at least once) is where there is an assurance that the message is sent to the destination at least once. The sender keeps on resending the message until a PUBACK confirmation is received. QoS 1 may lead to duplicated transmission but it is more reliable than QoS 0. The most trusted one is QoS 2 (exactly once). It makes use of a four-step handshake with PUBREC, PUBREL and PUBCOMP messages to ensure that a message is sent exactly once. Removing duplicates, QoS 2 increases delivery accuracy, however, it also increases the complexity of protocols and energy consumption [13, 15, 28].

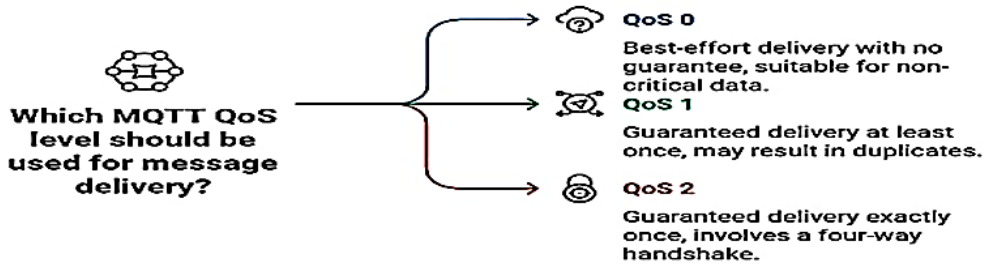

Fig. 3. MQTT QoS levels

The attributes of each QoS level make it suitable for specific use cases, as indicated in Table 1. That is QoS 0 minimizes latencies and overheads but poses the danger of message loss; QoS 1 is more reliable and can cause duplicates; QoS 2 is most accurate in delivery but uses more processing and memory. Choosing between the QoS levels thus involves a trade-off between the tolerance of an application to data loss, duplication, latency, resource consumption and energy consumption [15, 28].

Table 1. Comparison of MQTT QoS levels

QoS Level	Name	Delivery Guarantee	Duplicate Messages?	Use Case
0	At most once	No guarantee (message may be lost)	No	Low-priority data like sensor readings where loss is acceptable
1	At least once	Guaranteed delivery, but possible duplicates	Yes	Most IoT scenarios (e.g., temperature alerts, smart home commands)
2	Exactly once	Guaranteed delivery with no duplicates	No	Critical data (e.g., financial transactions, medical alerts)

Despite MQTT's flexible QoS framework, several practical limitations in IoT deployments diminish the utility of these levels. Unreliable wireless networks, especially those based on Long Range (LoRa) or ZigBee, are easily interfered with, collide and degrade their signals, resulting in retransmission or even entire messages being lost [39]. The energy limitations of most IoT nodes compound these difficulties. Battery-operated devices are frequently in sleep mode and multiple acknowledgments or multistep QoS handshakes, especially those involved in QoS 2, can quickly deplete energy supplies and reduce the lifetime of a device [34, 39]. Scalability also becomes a problem in the case of high device density. As the number of connected devices expands, MQTT brokers may struggle to manage subscriptions and forward messages efficiently, potentially leading to performance degradation under heavy load [30, 40, 41]. This limitation is often observed in environmental monitoring systems such as temperature sensing, where the unstable wireless connections have been proven to result in a lot of message loss and delay, which in turn directly influences QoS assurances [42]. To alleviate such scalability constraints, some researchers advocate edge-based MQTT broker architectures that distribute processing geographically and thus reduce centralized bottlenecks [43]. In addition, real-time applications such as industrial monitoring or healthcare require delivering messages in time. The current QoS mechanisms are static and reactive and do not respond to the changing network conditions in a short time frame, nor do they prioritize the latency-sensitive traffic leading to unacceptable delays in critically important services [25, 44].

3. Methodology of the Review

The current literature review is presented based on the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) guidelines to identify the relevant studies related to the MQTT protocol in IoT applications. An extensive search was conducted in five large databases IEEE Xplore, ScienceDirect, Springer, MDPI, and the ACM Digital Library. The search was intentionally limited to the period 2019-2025, and the number of initially collected references was 342 records from the mentioned databases with an additional 18 records from other sources. After the removal of 71 duplicates, 289 records were filtered using the title and abstract, which led to the appraisal of 155 full-text articles as eligible. This left 70 articles out due to the absence of empirical data (n = 28), irrelevance to the research topic (n = 24), methodological impossibilities (n = 11), and unavailability of the full-text (n = 7) which narrowed down to the inclusion of 85 articles in the qualitative synthesis. The studies included in the study were categorized into four major areas of research performance analysis (n = 28), quality-of-service mechanisms (n = 19), security implementations (n = 16) and practical applications (n = 22), as presented in Fig. 4.

4. QoS Enhancement Techniques

Recent research into MQTT focused on modifying its static QoS model by introducing enhancement techniques across multiple layers of the protocol architecture. Such efforts can be classified into five principal categories: improvements at the protocol level, network-aware adaptations, subscriber-driven cooperation, broker-side optimization, and adaptive QoS mechanisms as shown in Fig. 5 that respond to context or learned behavior. All the categories have different strategies to increase the reliability, efficiency, and scalability of IoT deployments.

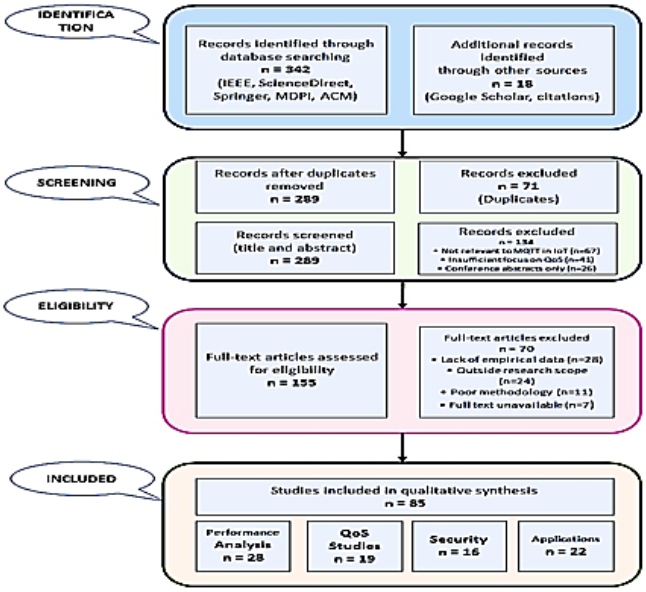

Fig. 4. PRISMA flow diagram

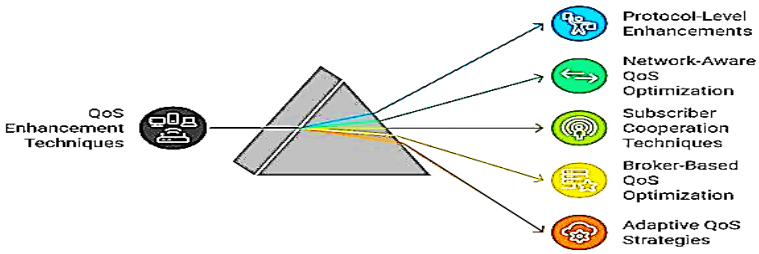

Fig. 5. QoS Enhancement techniques in the MQTT protocol

4.1. Protocol-level enhancements

Improvements at the protocol layer ought to improve the delivery of messages, and not change the overall architecture. Other works, as shown in Fig. 6, suggest that lightweight changes to the QoS 2 mechanism can be made by decreasing the handshake sequence without sacrificing delivery guarantees; simplified acknowledgment strategies, which are meant to reduce the overhead of the transmission [10, 13, 30].

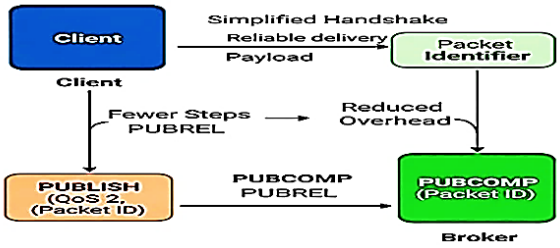

Fig. 6. A lightweight QoS 2 mechanism

Other solutions redefine or extend QoS control flags to allow more precise granularity of reliability, e.g., per-topic or per-session reliability modes. Also explored are hybrid models, which are a combination of both acknowledgment patterns of several QoS levels, like using QoS 1 with selective duplicate suppression to provide a lower latency and minimal message loss [15, 26, 45]. These modifications typically preserve compatibility with existing MQTT brokers and clients. They are however normally constrained by the simplicity of the design of the protocols and the deficiency of cross-layer cognizance. Moreover, protocol-level enhancements additionally integrate secure authentication mechanisms such as blockchain-based one-time-password (OTP) schemes, which may enhance trust without compromising MQTT's lightweight nature [46]. Other related proposals present extendible message formats that support secure thing-to-thing communication and are still transparent and interoperable [47].

4.2. Network-aware QoS optimization

A range of network-adaptive techniques has emerged for modulating the behavior of the MQTT protocol in response to communications environments. A common mode is adaptive retransmission schemes whereby the decision to retransmit a packet is made based on some metrics

like packet-loss rates or channel congestion. Such mechanisms often depended on signal-strength indicators or feedback received at the Media Access Control (MAC) layer to determine the quality of the link [25]. Additionally, researchers have examined adjusting MQTT's QoS levels to properties of distinct wireless technologies. Incidentally, the prioritization of packets sent on the IEEE 802.15.4 or Narrow Band-Internet of Things (NB-IoT) channels as either stringent or relaxed has been shown to improve delivery in congested environments [48]. Such prioritization often requires integration with cross-layer or hybrid protocol gateways to align with the MQTT message flow [36]. At the same time, congestion-sensitive routing strategies used by gateways have been utilized to limit upstream retransmission and load sharing among network routes [43, 49]. Moreover, hybrid gateway implementations that combine MQTT with CoAP protocols have shown success in adjusting message-handling strategies according to network congestion and endpoint characteristics [50]. Empirical evaluations comparing communication protocols within embedded-system constraints further establish that MQTT's QoS performance may degrade in unmanaged networks, thereby underscoring the necessity for adaptive mechanisms [51]. Even though these methods may be used to increase the responsiveness of systems in varied environments, they often require modifications to the network stack or coordinated communication with lower-level protocols and thus reduce interoperability and increase the complexity of implementation.

4.3. Subscriber cooperation techniques

Rather than relying exclusively on broker support, certain scholars advocate for the implementation of subscriber-level QoS enhancement strategies. As shown in Fig. 7, examples include redundant message reception, in which multiple subscribers simultaneously receive a message and may redistribute it through local retransmission mechanisms if packet loss is detected; local retransmission, wherein nodes that observe packet loss retransmit data locally; and neighbor-based recovery, wherein subscribers cooperate to repair disrupted connections [52-54].

A number of approaches also involve message caching at the subscriber level. In topic-based deployments, subscribers temporarily store incoming messages and proactively fulfill missing data requests from peers within the same group. This process is especially applicable in usage with mesh or cluster deployment with a high level of physical proximity that permits low-latency connections [55, 56]. Although the resilience can be enhanced by the use of cooperative techniques, it introduces new factors regarding the trust, security and synchronization in the case of heterogeneous or mobile configurations. Cooperative retransmission has been successfully applied in smart agriculture in soil humidity monitoring systems to provide integrity on data being conveyed over unreliable links [54]. Similar improvement has been demonstrated in maritime environments, where MQTT over LoRa has enabled multi-node subscriber collaboration to sustain connectivity across wide and harsh oceanic networks [56].

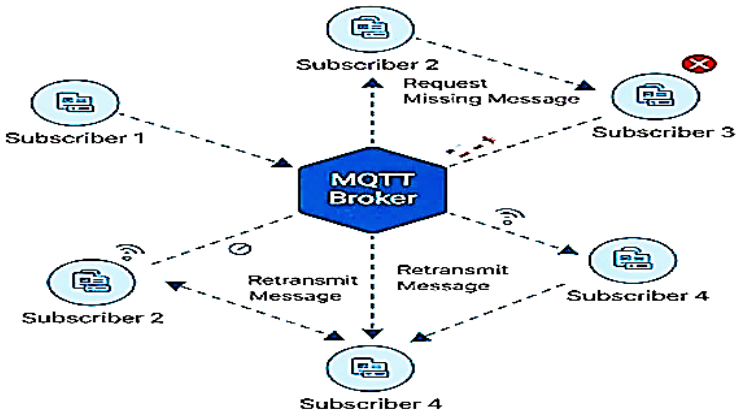

Fig. 7. Subscriber level collaboration

4.4. Broker-based QoS optimization

MQTT brokers serve as the fundamental nodes of message exchange in IoT networks. Recent studies have attempted to enhance their intelligence with the addition of features like message prioritization, selective buffering and context-aware scheduling, all meant to maximize the capability of traffic management [40, 41, 57]. Several systems have experimented with broker-side retransmission policies, in which the broker maintains messages (particularly under QoS levels 1 and 2) until successful acknowledgment is obtained; this practice eases the load on publishers. Moreover, distributed broker networks have been studied in the literature where the client-broker assignments are adapted dynamically to meet some objectives on load, locality, or reliability [41, 58]. Further analysis of broker infrastructure also sheds more light on their role in scalability and reliability in the presence of a complex IoT deployment. As an example, the Erlang Message Queue X (EMQX) architecture has been demonstrated to possess significant load distribution capabilities and minimize delivery loss among the topic-based topology [58, 59]. Separately, studies of MQTT performance within cloud-integrated architectures suggest that broker optimization is paramount to sustaining dependable throughput under large-scale message streams [60]. Taken together, these enhancements markedly improve scalability and reliability across MQTT networks; however, they often necessitate significant modifications to broker architectures, most notably in cloud-based and federated deployment models.

4.5. Adaptive QoS strategies

Adaptive QoS approaches actively manage the reliability of messages in response to the contextual parameters such as the stability of the network, battery level of the device and application priority. The suggestions exploited context-aware policies, in which the sensors that are prone to battery depletion decide to compromise their QoS to reduce power consumption [10, 61]. The more recent developments use machine learning to predict the best possible QoS settings. Historical datasets allow systems to calibrate QoS per message to maintain reliability and minimize resource usage. The models, in particular, deep neural networks and reinforcement learning (RL) agents, have been shown to predict message failures or network congestion and adapt to it [24]. In addition, deep learning techniques are effective in the solution of QoS trade-offs

within real-time communication systems; the neural models can re-adjust parameters during run-time, provided that sufficient computational resources are available depending on the dynamics of the network [24]. Although adaptive schemes are highly beneficial (as shown in Fig. 8), they are usually faced with deployment challenges, such as the need to have substantial training data, computing facilities, and the inability to withstand real-time system constraints. To address these obstacles, datasets such as the MQTT set have been introduced, affording reproducible training and assessment for intelligent QoS adaptation [62].

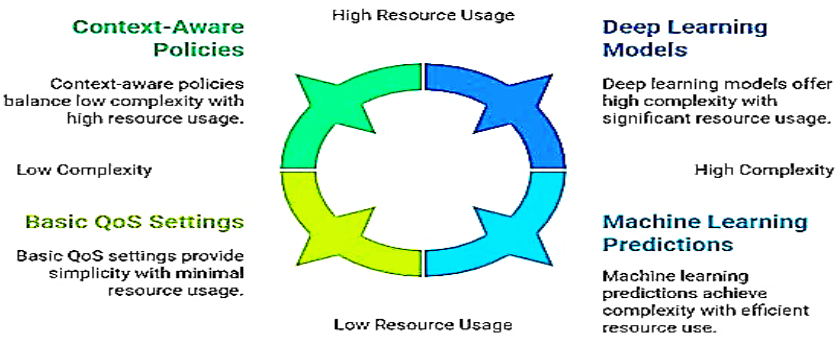

Fig. 8. Adaptive QoS strategies

5. Comparative Analysis of QoS Techniques

This section makes a rational comparison of QoS enhancement options in MQTT in three major dimensions: performance (in terms of packet loss, latency, throughput and energy consumption), protocol complexity and overhead (in terms of processing, memory, message size) and compatibility with most popular use cases such as smart home, healthcare, industrial IoT and smart agriculture. The aim of this well-organized analysis of this nature is to provide a firm presentation of the involved trade-offs and to help in making informed decisions about particular IoT implementations.

5.1. Performance metrics

To conduct a critical review of methods of QoS improvement of MQTT-based IoT systems, it is necessary to measure indicators of performance in a systematic way. Packet loss, latency, throughput and energy consumption are the main metrics used to evaluate system reliability, responsiveness, and long-term sustainability particularly in resource-constrained scenarios [36].

5.1.1. Packet loss

Packet loss, or Packet Error Rate (PER), in IoT refers to the ratio of lost or corrupted data packets to the total packets transmitted over a network. It is a critical metric for assessing communication reliability in IoT systems, especially in low-power, wireless environments [63]. This study showed that the suggested flow control mechanism reduced the packet drop to 98% compared to the standard MQTT implementation. The reduction of PER is the most significant objective of QoS enhancement. The study of Enhanced MQTT-SN in [36] improves average packet delivery by 30% during publish-subscribe operations. Techniques like redundancy of messages and retransmission based on brokers have been observed to attain a significant reduction in the value of PER. An example is that of Zunino [30] who demonstrated that adaptive redundancy mechanisms lowered the loss of messages during high network congestion. Furthermore, machine-learning-based models proposed by [64] are capable of identifying failure patterns in network behavior and adjusting MQTT message routing proactively, thereby improving reliability and reducing packet-loss rates in dynamic IoT environments. Proximity-based designs that take advantage of the proximity of edges, such as the edge-based architecture proposed by [43] make use of the geographical proximity to support local retransmission and achieve higher delivery reliability in high-density deployments.

5.1.2. Latency

In the context of IoT, latency refers to the time delay between the moment when a device sends data (e.g., a sensor measurement) and the moment when the system processes or acts on it [65]. Latency is an important performance parameter of a communication system with strict time requirements, e.g., healthcare and industrial control. The work in [10] showed that simplifying the QoS 2 handshake process is able to decrease the round-trip message latency by a large margin through optimization of the handshake protocol. Also [25] further demonstrated that MQTT variants, among them PrioMQTT can significantly reduce queueing delay from 51.30% to 79.14% compared to standard MQTT through prioritization of emergency messages. However, latency improvement may require energy-aware mechanisms that dynamically tradeoff between speed of message delivery and resource consumption [61]. This simulation study showed that the given algorithm has an advantage of 7.41% in the weighted sum of the packet loss ratio in comparison to the traditional one. Moreover, the algorithm is discovered to consistently select near-optimal strategies as the signal-to-noise-ratio threshold varies. Moreover, multiprotocol hybrid configurations, exemplified by MQTT-CoAP integrations proposed by [66], and message prioritization [67] are found to achieve lower end-to-end latencies.

5.1.3. Throughput

Throughput refers to the amount of data successfully transmitted over a network per unit of time. It measures the effective data transfer rate, accounting for protocol overhead, retransmissions, and network congestion [68]. Extensive literature on MQTT brokers indicates that throughput improvements stem from optimizing broker-side operations. Distributed broker architectures, as described in [57], spread the message load over multiple nodes and thus increase the throughput at heavy traffic load conditions. Throughput can be improved by enhancing channel usage through mechanisms that decrease the retransmissions like adaptive acknowledgments [69]. Empirical studies by [59] with

MQTT on Raspberry Pi-based broker platforms showed that broker performance is strongly influenced by client density, with refined broker logic improving message throughput.

5.1.4. Energy consumption

Within battery-powered IoT nodes, energy efficiency is a critical element. QoS modifications that reduce the frequency of retransmissions, for example, the MQTT User Properties (MUP) framework introduced by [45] directly curtail energy consumption. Investigations by [70] and [55] demonstrated that integrating LoRa with MQTT allows appreciable energy savings by coupling low-power transmission with reduced acknowledgment frequencies. To mitigate the energy consumption related to the overhead imposed by the MQTT protocol operations, [34] examined the performance of digital signatures. It was found that lightweight cryptographic approaches can safeguard security while constraining energy expenditure. Adaptive QoS downgrading associated with the battery-level indicator suggested by [61] extends the life of operation without significantly reducing reliability and the algorithm has an advantage of 49.49% in energy consumption in its simulation study. Table 2 presents a comparison of key MQTT-enhancing techniques in terms of the described performance metrics.

Table 2. MQTT QoS performance metrics

Study / Approach	Packet Loss Reduction	Latency Impact	Throughput Impact	Energy Efficiency	Notes / Special Features
Adaptive Seamless Redundancy (ASR) [30] (2024)	Reduced from 9.6% (single path) to ~0.02%–0.2% (ASR) in critical conditions	Slight improvement in normal conditions; slight increase in critical conditions	implied decrease due to redundancy	Not discussed	Adaptive redundancy
Adaptive QoS (Epsilon-Greedy) [61] (2023)	Dynamic trade-off (near-zero to high based on QoS mode)	Not explicitly measured	Not explicitly measured	Improved by 49.49% vs. fixed strategies	Uses epsilon-greedy MAB (Multi-Armed Bandit)
Edge-Based MQTT Architecture [43] (2020)	improved (due to reduced broker load)	reduced (due to efficient message routing)	Improved (reduced broker overload)	improved (less overhead)	Geo-aware MQTT edge architecture
Adaptive QoS Control for MQTT-SN [10] (2022)	Improved hitrate (up to ~100% with QoS 2)	Reduced or controlled based on mode	Not explicitly measured	Not explicitly measured	- Centralized QoS controller - Three operational modes: Latency-constrained, Hitrate-constrained, Autonomous - No client-side intelligence required - Uses network simulation (NS-3) - Focus on subscriber-side QoS
Priority-based MQTT (PrioMQTT) [25] (2024)	Not explicitly measured	Reduced Round TripTime (RTT) by (51–79) % vs. standard MQTT	Not explicitly measured	Not explicitly measured	- UDP (User Datagram Protocol)-based for low latency - 64-bit priority field in MQTT header - Backward compatible with standard MQTT - No special hardware required (commercial-off-the-shelf (COTS)-friendly)
Smart Irrigation Topic Logic [67] (2020)	Not explicitly measured	Not explicitly measured	Not explicitly measured	Improved (efficient water use)	Uses MQTT with a Raspberry Pi broker - Node MCU + sensors (DHT-11, moisture) - ThingSpeak cloud + WEKA for prediction - Low-cost design for farmers
MUP: Secure MQTT Updates [45] (2020)	Uses application-layer slicing and a Stop-and-Wait protocol for reliable delivery over constrained networks.	Increased by the two-phase process (manifest check first) and slicing overhead. Necessary for security.	Low due to constrained device limits and fragmentation. Optimizing slice size is critical.	High. The two-phase approach prevents downloading invalid firmware, saving significant energy.	Focus on Secure over-the-air (OTA) updates for constrained devices. Finding: Integrates secure, energy-efficient updates into the MQTT architecture

Algorithms using machine learning have a high potential to improve throughput and decrease packet loss and, in some environments, also offer energy savings. Nevertheless, all these advantages are accompanied by minor increments in the complexity of calculations because of the necessity to make real-time decisions. On the contrary, lightweight protocol-level changes provide low latency and low implementation costs, thus making them appealing to resource-limited devices. Time-sensitive cases can be made more responsive by optimizations at the broker, but will be more-costly in energy in event-driven cases.

5.2. Protocol complexity

5.2.1. Memory and CPU requirements

The trade-offs associated with computations tend to be experienced when QoS is improved. Message buffering, session tracking, and ML-inference [71] mechanisms have the effect of raising the memory footprint required. Also in [36], the Enhanced MQTT-SN reduces memory usage by 5.6MB during publish-subscribe operations. Recent reviews on embedded platforms [72] showed that the implementation of such measures as message duplication checks or queue management may increase the sensor network consumption depending on the frequency of

the messages and the loading of clients. The storage and runtime inference cost of such a design also must be taken into consideration in devices that implement intelligent agents or over-the-air learning, e.g., a machine learning-based models framework by [64], making them unfeasible to execute on microcontrollers with limited memory budget such as the ESP8266 or Arduino platforms as presented in Fig. 9.

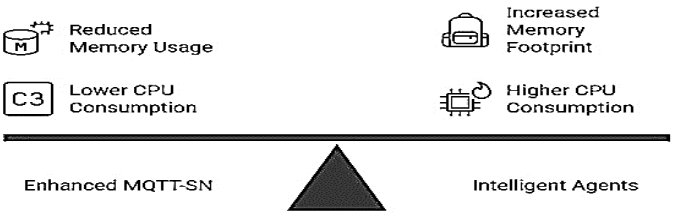

Fig. 9. Balancing memory and CPU in MQTT implementations

5.2.2. Message size overhead

A substantial body of literature proposes QoS mechanisms that embed metadata within the MQTT payload. Among these approaches, MUP [45] and CoAP-MQTT-hybrid formats [73] append extra headers for encryption, retries, or context information. Though the resulting message-size overhead is insignificant in Ethernet-based infrastructures, it is significantly larger in narrowband networks, like NB-IoT, as Khan and Pirak showed empirically [48]. Mechanisms that integrate MQTT with blockchains [74] and [75] also incur message overhead, thereby constraining their applicability in resource-limited settings. On the other hand, the ones that reduce QoS by reducing the number of packet exchanges rather than enlarging message size, e.g., adaptive handshakes [10], trade off performance and resource use better. The complexity of contemporary MQTT QoS protocols is thus summarized in Table 3. Table 3 indicates that some of the QoS enhancement methods have high memory and processing resource requirements, especially when combined with more advanced security features or machine-learning frameworks, thus limiting them to be used on resource-limited embedded systems. On the other hand, those that have low computational complexity require low memory usage and CPU and produce small message sizes, thus fitting the requirements of narrowband and feature-limited networks. In this context, the choice of a solution that skillfully balances implementation complexity with the benefits that can be achieved is a critical factor, particularly in the case of large-scale IoT environments.

Table 3. MQTT QoS protocol complexity aspects

Study / Technique	Memory Overhead	CPU Load Impact	Message Size	Implementation Complexity	Notes / Constraints
(MUP) [45] (2020)	27.49 KB (device desc.) + buffers	High (Elliptic Curve Digital Signature Algorithm (ECDSA) verify + Secure Hash Algorithm (SHA)-256)	Manifest + 220-880 B slices	High (app-layer slicing, flow control)	Pre-shared key. 6LoWPAN. 81.54 s update time. Partial MQTT v5. Two-phase
Adaptive QoS + Power Control [61] (2023)	Not Specified	Not Specified	Not Specified	Moderate	MAB /ML-based optimization using historical data
Lightweight Intrusion Detection [76] (2019)	Moderate	High (real-time tree matching & similarity computation)	Normal MQTT messages	High (requires process tree construction & behavioral model training)	Detects subtle anomalies via efficient partial process tree checking
Blockchain-Based OTP Authentication [75] (2020)	Moderate (hashing, Ethereum interactions)	High (cryptographic operations, blockchain transactions)	Normal MQTT + small OTP data	High (Ethereum integration, smart contract development)	Ethereum-dependent, no TLS, privacy via hashing
Proposed (Blockchain Sharding) [74] (2023)	~175 MB	~10%	~6000 Kbps (QoS-2)	High (smart contracts, sharding logic)	More suitable for resource-constrained environments. Enhances security and privacy without a central authority
MQTT + CoAP (Hybrid) [73] (2019)	Moderate (depends on encapsulation)	Moderate (combined overhead)	Flexible (depends on use case)	High (protocol translation, URL (Uniform Resource Locator) handling)	Good for balanced priority updates. Increased complexity
Adaptive QoS MQTT-SN [10] (2022)	Low (controller on broker)	Low (lightweight control logic)	Adaptive (based on QoS level)	Moderate (requires broker integration)	Dynamically adjusts QoS (0,1,2) based on real-time network conditions (PER, latency)
Distributed MQTT Broker [57] (2023)	Moderate (due to Overlay Routing Tables (ORT), Publication Routing Tables (PRT), Subscription Routing Tables (SRT), topic - ORT tables)	High (especially under stress, +22% CPU usage)	Variable (efficient routing reduces redundancy)	High (requires broker modification, dynamic overlay management)	- Supports dynamic broker discovery - Topic-based overlay routing - Recovery time: ~59.1s - Convergence time: ~19.5s - Signaling overhead increases with topic count (e.g., +1500% for 60 topics)
Digital Signature Analysis [34] (2020)	Varies by device class	High for crypto operations	Increased (due to signature)	Moderate to High	- level 2: Best crypto performance - level 1: Fastest message delivery - level 0: Most constrained

5.3. Applicability to use cases

The suitability of the classical MQTT protocol and its variants depends on the specific requirements of each application. An application may comprise multiple sub-functions, each with distinct needs. The following section discusses the use of the MQTT protocol in major, widely adopted IoT-based applications as depicted in Fig. 10.

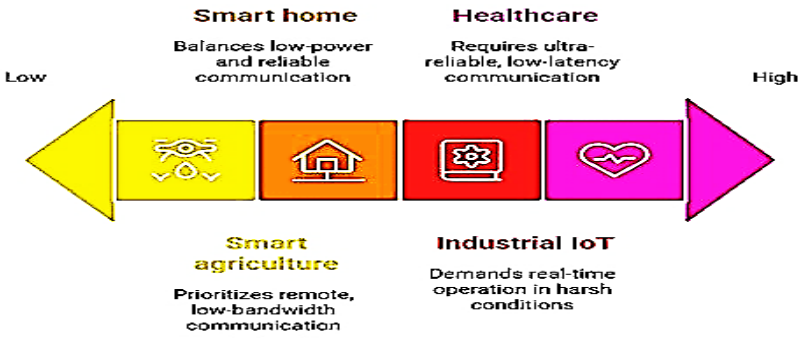

Fig. 10. IoT application requirements range from low to high performance

5.3.1. Smart home

In smart home applications, MQTT is frequently employed for event-driven load switching control and security. In this area, latency with moderate reliability and QoS is acceptable. In most cases, the availability of energy, equipment, and device resources is not a limiting factor. Some techniques have been shown to work rather well, including priority-aware scheduling [25] and traffic shaping at the broker side [57]. At the same time, security-conscious mechanisms, such as the one proposed by [77] allow secure transmission and responsiveness to be maintained.

5.3.2. Healthcare

Healthcare applications have QoS requirements that support high-priority alarms as well as constant background surveillance. Adaptive models like 5G-smart healthcare solution proposed by [14] are adaptive to QoS with respect to the importance of the data. Meanwhile, redundancy and delay-sensitive optimizations, as illustrated by Younas et al. in IoT Edge Devices Driven Healthcare [78] serve as an essential component in medical infrastructures.

5.3.3. Industrial IoT

In the case of IIoT, there is a need for deterministic behavior and resiliency of the system under different operation conditions. It can be allowed to operate under a high load, due to strong broker clustering [4] and enhanced security protocols [18, 79]. For Industrial environment a study like [80] discussed IoT technology improving automation, but highlighted concern about older systems still using outdated devices. It introduced a cost-effective IoT gateway prototype that connects Modbus RTU (Remote Terminal Unit) sensors to an MQTT cloud, allowing for two-way communication. The prototype was tested successfully in a lab setting.

5.3.4. Smart agriculture

Agricultural systems are energy-conscious and environmentally flexible. Integrations of MQTT with low-power wireless protocols (e.g., LoRa, BLE) have proven advantageous [55, 70]. Real-world deployments, including hydroponic farms equipped with MQTT brokers [81] and mushroom cultivation monitoring systems [82], demonstrated the viability of lightweight QoS strategies in remote, infrastructure-scarce settings. A summary of the MQTT QoS protocol applicability across these main use cases is provided in Table 4. Table 4 provides a summary of the significant variations in the efficiency of various QoS enhancement methods in covering specific areas of applications. In smart home and healthcare, strategies that emphasize a strict latency reduction and the delivery of important messages are beneficial because temporal precision is essential. The energy-conservation approaches and energy-network operation under connectivity constraints are more appropriate in smart agriculture and remote or underserved areas. In industrial use cases, stability of the system under a high network load is still a requirement, but the stability might require more architectural complexity.

6. Limitations and Challenges

The advancement of QoS in MQTT has generated notable progress in addressing specific constraints within Publish/Subscribe systems. Nevertheless, several limitations and critical challenges continue to impede the realization of more intelligent, resilient, and scalable MQTT-based IoT deployments in real-world scenarios.

6.1. Limitations of current cooperative subscriber approaches

The current decentralized recovery mechanisms employ message caching, local retransmission, and redundancy sharing. Although they are useful, they are usually poorly consistent and synchronized especially in mobile or heterogeneous networks. Peer-based recovery algorithms [53, 55] assume that the neighborhood relationships are stable, which is hard to maintain in settings with dynamic topologies.

6.2. Lack of standardized support for advanced QoS in MQTT v5

Although MQTT v5 has been enriched with user properties, topic aliases, and reason codes, it still lacks standardized support for adaptive or intelligent QoS mechanisms. Even most advanced capabilities (such as priority tagging [18]) are implementation-specific and incompatible

between brokers. Moreover, MQTT v5 does not offer built-in mechanisms for context-aware QoS negotiation, inter-layer coordination [57], or energy-based QoS adaptation

6.3. Need for real-world evaluation and datasets

A significant proportion of QoS research in MQTT is still confined to simulation studies or constrained testbeds. There are only a limited number of studies that report experimental results with real deployments or resource-limited hardware, like [59] and [34]. Such a deficiency in standardized datasets and reproducible evaluations makes it hard to compare the competing techniques. Furthermore, most of the research does not take into account multi-client scaling, mobility, or intermittent connectivity, which are common in working IoT systems [76, 83].

6.4. Security and privacy concerns in cooperative QoS

The cooperative retransmission among multiple subscribers introduces a set of security vulnerabilities, exemplified by spoofing, eavesdropping, and replay attacks. Because messages may be stored, or forwarded by the participating devices, issues of trust establishment, authentication, and data integrity gain particular relevance. Partial mitigations are also possible with technologies like blockchain-based authentication [74, 75], but they are associated with latency and a high processing cost. Furthermore, subscriber-level security models remain less mature than broker-side defenses [79, 84].

6.5. Integration with edge computing and fog nodes

Edge and fog computing platforms present opportunities for localized processing and QoS control, but their integration with MQTT remains fragmented. Learning-based QoS models have been proposed by [64] and [71], however, they are limited in terms of the hardware requirements of deployment on edge devices.

Table 4. MQTT QoS protocol applicability to main use cases

Study / Technique	Smart Home	Healthcare	IIoT	Smart Agriculture	Comments / Adaptability	Quantitative Insights
PrioMQTT – Prioritized MQTT [25] (2024)	Excellent	Suitable for alerts	Excellent	Moderate	Effective for event prioritization	RTT reduction up to 79.14%; MLR (Message Loss Rate) < 10^{-7}; Median queue size < 7; supports 64 priority levels
(5G-smart healthcare) [14] (2020)	Moderate	Excellent	Moderate	Low	Tailored for e-health platforms	5G + IoT healthcare synergy; latency goal: ≤1ms; reliability ≥99.999%
Advanced MQTT Broker [57] (2023)	High	Moderate	High	Low	Designed for scalable, fault-tolerant distributed broker networks. Optimal for dynamic, multi-topic environments.	Convergence time: ~19.5s, Repair time: ~59.1s, Traffic overhead increases with topics (30% to 1500% based on topic count).
EMMA (Epsilon-greedy based MQTT QoS & Power Control) [61] (2023)	Moderate	Low	Excellent	Low	Optimized for resource-constrained IoT in industrial settings (e.g., power distribution). Dynamically balances packet loss and energy consumption. Not designed for low-latency or high-priority use cases.	Reduces the weighted sum of packet-loss + energy by 7.41–49.49% vs. fixed strategies. Supports dynamic channel adaptation.
(LoRa-based MQTT) [55] (2020)	Low	Low	Moderate	Excellent	Best for low-data, long-range, off-grid monitoring. Not for real-time control.	Range: 3-10km+ Delay: 500-3000ms+
(Hydroponic MQTT Deployment) [81] (2019)	Excellent	Excellent	Excellent	Excellent	Specialized security for any system using MQTT. Server-based, lightweight for devices.	Accuracy: ~99.9% Focus: MQTT attacks Features: 31
(Priority-enabled MQTT) [18] (2024)	Excellent	Excellent	Excellent	Good	Ideal for any system requiring reliable, low-latency emergency messaging. Adds complexity with queue management.	Queues: 3 (Urgent, Critical) Performance: Lower delay/jitter for high-priority queues
(MQTT-Modbus Gateway) [80] (2019)	Good	Moderate	Excellent	Good	Essential for integrating legacy industrial equipment (Modbus) into modern IoT (MQTT) systems. A bridge between old and new.	Protocols: Modbus RTU ↔ MQTT Function: Bidirectional command/alert mapping Hardware: Raspberry Pi
Digital Signature Analysis [34] (2020)	Moderate	High	Moderate	Low	Security-focused, moderate overhead	- Increases data size significantly. - Slows down message processing. - Raises energy consumption.
Redundant Delivery Approach [30] (2024)	Good	Excellent	Excellent	Good	Ideal for any critical application requiring ultra-high reliability and low latency over unreliable networks (e.g., Internet). Adds significant complexity.	Redundancy: 2+ paths Performance: ~99.99% success rate, lower latency vs. single path Method: RL-based adaptive path selection

7. Future Directions

In current MQTT-based IoT systems, limitations in existing QoS enhancement techniques allow further investigation into solutions that are more adaptive, intelligent, and interoperable. This section outlines possible directions for the design of next-generation MQTT architectures.

7.1. Intelligent broker design for QoS adaptation

Future MQTT brokers should be redefined as context-aware decision entities capable of adjusting QoS levels dynamically. Brokers can apply intelligent load-balancing, retry, or prioritization algorithms by monitoring client behavior and network conditions as well as message properties. The previous analysis of [57] and [4] defined scheduling and buffering procedures at the broker side, but these procedures are fixed. Adding real-time analytics and predictive modelling at the broker level would have significant gains in reliability and scalability.

7.2. Subscriber-level collaboration frameworks

A systematic framework is needed for collaborative behavior among subscribers, such as message caching, peer-to-peer retransmission, and group acknowledgement, which could mitigate dependency on the broker. This may be assisted in terms of protocol extensions or side-channel signaling (e.g., multicast feedback). Potential applications are shown in [53] and [55], but these still need to be improved in synchronization, duplicate-message suppression, and security.

7.3. AI/ML-based QoS prediction and control

It is a possibility, but has not been much explored, to apply machine-learning techniques to QoS control. The historical network condition, the device behavior, or context can be used by the learning models to predict the optimal QoS parameters. The works in [71,61,64] proposed a DNN and ML-based functions, but more efforts are needed to develop lightweight versions that can run in real-time on limited edge devices.

7.4. Cross-layer QoS-aware protocol stacks

Current QoS management solutions are mostly restricted to the application layer, and therefore, they are not always very effective. Architectures based on cross-layer coordination would allow real-time parameter updating in the transport, network, or MAC layers.

7.5. Interoperability with other protocols

In heterogeneous IoT environments, interoperability between MQTT and related protocols such as CoAP, LwM2M, or Extensible Messaging and Presence Protocol (XMPP) becomes essential. Translation layers and hybrid gateways proposed in [73] and [66] need further extension toward more general interoperability frameworks.

7.6. Testing in high-mobility or high-density scenarios

The prevailing assessments of QoS methods are carried out in either static or homogeneous environments. Mobile nodes (vehicles, wearables) or dense infrastructures (smart cities, factories) are common characteristics of real IoT deployments, which present new challenges with respect to handover latency, broker reassignment, and group synchronization. The limited literature in this context [85] could be extended to consider mobility traces and different topologies to explore scalability and robustness of the system in real-life environments.

8. Conclusions

The main aim of this review was to systematically identify, analyze, and compare QoS enhancement techniques in MQTT-based IoT systems, with particular emphasis on emerging cooperative and application-aware mechanisms. The three fixed levels of native QoS mechanisms cannot fit the dynamic, heterogeneous needs of contemporary IoT deployments. To address this shortcoming, scholars suggested a spectrum of enhancement strategies, which the present paper organizes into five broad categories: protocol-level modifications, network-aware adaptation, subscriber cooperation schemes, broker-based optimizations, and QoS adaptive solutions that incorporate context awareness or machine learning. The categories possess their advantages and disadvantages. Changes at the protocol level are not very flexible or lightweight. Network-aware and subscriber-cooperation techniques yielded enhanced scalability and reliability but introduced additional complexity. Context or machine learning-based QoS solutions on the other hand can be fine-grained controlled at the cost of higher computing requirements and the need to have more powerful data models. By organizing the literature into a structured taxonomy and evaluating techniques across different QoS dimensions and IoT application domains, this study provided a unified view of how reliability, latency, and delivery guarantees can be improved beyond the standard MQTT protocol operations. The significance of this review lies in highlighting the shift from broker-centric QoS control toward collaborative, subscriber-assisted, and adaptive strategies, which are increasingly necessary in dynamic, large-scale, and resource-constrained IoT environments. Through comparative tables and cross-domain analysis, this work offered both researchers and practitioners a clear foundation to figure out the challenges, trade-offs, and future perspectives for better selecting, designing, and further developing effective QoS enhancement solutions for real-world MQTT deployments.

Acknowledgment

The authors gratefully acknowledge the support provided by the Technical Engineering College, Northern Technical University, Kirkuk, Iraq.

References

[1] G. Beniwal and A. Singhrova, “A systematic literature review on IoT gateways,” J. King Saud Univ. – Comput. Inf. Sci., vol. 34, no. 10, pp. 9541–9563, Nov. 2022. https://doi.org/10.1016/j.jksuci.2021.11.007.
[2] M. Mansour et al., “Internet of Things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions,” Energies, vol. 16, no. 8, p. 3465, Apr. 2023. https://doi.org/10.3390/en16083465.
[3] A. Kirimtat, O. Krejcar, A. Kertesz, and M. F. Tasgetiren, “Future trends and current state of smart city concepts: A survey,” IEEE Access,

vol. 8, pp. 86448–86467, 2020. https://doi.org/10.1109/ACCESS.2020.2992441.
[4] J. Ali and M. H. Zafar, “Improved end-to-end service assurance and mathematical modeling of MQTT-based massively deployed devices in smart cities,” Alexandria Eng. J., vol. 72, pp. 657–672, Jun. 2023. https://doi.org/10.1016/j.aej.2023.04.014.
[5] C. K. Metallidou, K. E. Psannis, and E. A. Egyptiadou, “Energy efficiency in smart buildings: IoT approaches,” IEEE Access, vol. 8, pp. 63679–63699, 2020. https://doi.org/10.1109/ACCESS.2020.2984461.
[6] C. D’Ortona, D. Tarchi, and C. Raffaelli, “Open-source MQTT-based end-to-end IoT system for smart city scenarios,” Future Internet, vol. 14, no. 2, p. 57, Feb. 2022. https://doi.org/10.3390/fi14020057.
[7] Y. Im and M. Lim, "E-MQTT: End-to-end synchronous and asynchronous communication mechanisms in the MQTT protocol," Appl. Sci., vol. 13, no. 22, p. 12419, Nov. 2023. https://doi.org/10.3390/app132212419.
[8] E. Longo, A. E. C. Redondi, M. Cesana, A. Arcia-Moret, and P. Manzoni, “MQTT-ST: A spanning tree protocol for distributed MQTT brokers,” in Proc. IEEE Int. Conf. Commun. (ICC), Dublin, Ireland, Jun. 2020, pp. 1–6. https://doi.org/10.1109/ICC40277.2020.9149046.
[9] B. Motamedi and B. Villányi, “A reliable publish–subscribe mechanism for IoT-enabled smart greenhouses,” Appl. Sci., vol. 14, no. 15, p. 6407, Jul. 2024. https://doi.org/10.3390/app14156407.
[10] F. Palmese, A. E. C. Redondi, and M. Cesana, “Adaptive quality of service control for MQTT-SN,” Sensors, vol. 22, no. 22, p. 8852, Nov. 2022. https://doi.org/10.3390/s22228852.
[11] H. Anwer, F. Azam, M. W. Anwar, and M. Rashid, “A model-driven approach for load-balanced MQTT protocol in Internet of Things,” in Advances in Intelligent Systems and Computing, vol. 993, Cham, Switzerland: Springer, 2020, pp. 368–378. https://doi.org/10.1007/978-3-030-22354-0_33.
[12] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, “A survey on the role of IoT in agriculture for the implementation of smart farming,” IEEE Access, vol. 7, pp. 156237–156271, 2019. https://doi.org/10.1109/ACCESS.2019.2949703.
[13] A. J. H. Hintaw, “Performance analysis of MQTT protocol in IoT environments: Impact of payload size and QoS on key metrics,” Tech. Romanian J. Appl. Sci. Technol., vol. 28, pp. 43–57, Mar. 2025. https://doi.org/10.47577/technium.v28i.12649.
[14] A. Ahad et al., “Technologies trend towards 5G network for smart healthcare using IoT: A review,” Sensors, vol. 20, no. 14, p. 4047, Jul. 2020. https://doi.org/10.3390/s20144047.
[15] B. Mishra and A. Kertesz, “The use of MQTT in M2M and IoT systems: A survey,” IEEE Access, vol. 8, pp. 201071–201086, 2020. https://doi.org/10.1109/ACCESS.2020.3035849.
[16] S. V. Mukherji, R. Sinha, S. Basak, and S. P. Kar, “Smart agriculture using Internet of Things and MQTT protocol,” in Proc. COMITCon, Faridabad, India, Feb. 2019, pp. 14–16. https://doi.org/10.1109/COMITCon.2019.8862233.
[17] M. Salagean and D. Zinca, “IoT applications based on MQTT protocol,” in Proc. ISETC, Timisoara, Romania, Nov. 2020, pp. 1–4. https://doi.org/10.1109/ISETC50328.2020.9301055.
[18] P. S. Akshatha, S. Divyashree, and S. M. Dilip Kumar, “Priority-enabled MQTT: A robust approach to emergency event messaging,” J. Eng. Appl. Sci., vol. 71, no. 1, p. 67, Dec. 2024. https://doi.org/10.1186/s44147-024-00400-2.
[19] S. Barkat, A. Bilami, and A. Benayache, “MQTT-based QoS model for IoT-M2M critical applications,” Int. J. Distrib. Syst. Technol., vol. 12, no. 4, pp. 47–67, Jan. 2022. https://doi.org/10.4018/IJDST.287862.
[20] X. Liu, T. Zhang, N. Hu, P. Zhang, and Y. Zhang, “Method of Internet of Things access and network communication based on MQTT,” Comput. Commun., vol. 153, pp. 169–176, Mar. 2020. https://doi.org/10.1016/j.comcom.2020.01.044.
[21] M. Quamara, B. B. Gupta, and S. Yamaguchi, “MQTT-driven remote temperature monitoring system for IoT-based smart homes,” in Proc. IEEE GCCE, Osaka, Japan, Oct. 2019, pp. 968–970. https://doi.org/10.1109/GCCE46687.2019.9015603.
[22] A. J. Hintaw et al., “MQTT vulnerabilities, attack vectors and solutions in the Internet of Things,” IETE J. Res., vol. 69, no. 6, pp. 3368–3397, Aug. 2023. https://doi.org/10.1080/03772063.2021.1912651.
[23] S. Kojima et al., “Enhancement of reliability of message delivery in many-subscriber MQTT networks,” in Proc. IEEE APCC, Bali, Indonesia, Nov. 2024, pp. 6–12. https://doi.org/10.1109/APCC62576.2024.10767914.
[24] S. Pawar et al., “Evaluation of quality of service parameters for MQTT communication in IoT applications using deep neural networks,” Int. J. Inf. Technol., vol. 16, no. 2, pp. 1123–1136, Feb. 2024. https://doi.org/10.1007/s41870-023-01664-2.
[25] G. Patti, L. Leonardi, G. Testa, and L. Lo Bello, “PrioMQTT: A prioritized version of the MQTT protocol,” Comput. Commun., vol. 220, pp. 43–51, Apr. 2024. https://doi.org/10.1016/j.comcom.2024.03.018.
[26] M. Domingues, J. N. Faria, and D. Portugal, “Dimensioning payload size for fast retransmission of MQTT packets under network disconnections,” EURASIP J. Wirel. Commun. Netw., vol. 2024, no. 1, p. 2, Jan. 2024. https://doi.org/10.1186/s13638-023-02327-3.
[27] E. K. J., “Optimizing data transfer speed and performance evaluation of MQTT in IoT,” J. Inf. Syst. Eng. Manag., vol. 10, no. 39s, pp. 231–244, Apr. 2025. https://doi.org/10.52783/jisem.v10i39s.7144.
[28] J. Toldinas et al., “MQTT quality of service versus energy consumption,” in Proc. IEEE Electronics, Palanga, Lithuania, Jun. 2019, pp. 1–4. https://doi.org/10.1109/ELECTRONICS.2019.8765692.
[29] D. Dinculeană and X. Cheng, “Vulnerabilities and limitations of MQTT protocol used between IoT devices,” Appl. Sci., vol. 9, no. 5, p. 848, Feb. 2019. https://doi.org/10.3390/app9050848.
[30] C. Zunino et al., “Adaptive seamless redundancy to achieve highly dependable MQTT communication,” IEEE Trans. Ind. Informat., vol. 20, no. 1, pp. 984–994, Jan. 2024. https://doi.org/10.1109/TII.2023.3271708.
[31] N. Saha et al., “Performance evaluation framework of MQTT client libraries for IoT applications in manufacturing,” Manuf. Lett., vol. 41, pp. 1237–1245, Oct. 2024. https://doi.org/10.1016/j.mfglet.2024.09.150.
[32] M. Martí, C. Garcia-Rubio, and C. Campo, “Performance evaluation of CoAP and MQTT-SN in an IoT environment,” in Proc. UCAmI, Nov. 2019, p. 49. https://doi.org/10.3390/proceedings2019031049.
[33] M. N. Jahantigh et al., “Integration of Internet of Things and cloud computing: A systematic survey,” IET Commun., vol. 14, no. 2, pp. 165–176, Jan. 2020. https://doi.org/10.1049/iet-com.2019.0537.
[34] A. Fauzan, P. Sukarno, and A. A. Wardana, “Overhead analysis of digital signatures in MQTT for constrained IoT devices,” in Proc. IC2IE, Yogyakarta, Indonesia, Sep. 2020, pp. 415–420. https://doi.org/10.1109/IC2IE50715.2020.9274651.
[35] P. Gupta and I. O. Prabha M., “A survey of application layer protocols for Internet of Things,” in Proc. ICCICT, Mumbai, India, Jun. 2021, pp. 1–6. https://doi.org/10.1109/ICCICT50803.2021.9510140.
[36] A. Mukherjee, N. Dey, and D. De, “EdgeDrone: QoS-aware MQTT middleware for mobile edge computing,” Comput. Commun., vol. 152, pp. 93–108, Feb. 2020. https://doi.org/10.1016/j.comcom.2020.01.039.

[37] K. Mekki, E. Bajic, and F. Meyer, "Indoor positioning system for IoT devices based on BLE and MQTT," in Proc. IEEE WF-IoT, Limerick, Ireland, Apr. 2019, pp. 787–792. https://doi.org/10.1109/WF-IoT.2019.8767287.
[38] R. A. Nathi and D. Sutar, "Object security scheme based on access policies using MQTT protocol," in Proc. IEEE ICCCNT, Kanpur, India, Jul. 2019, pp. 1–6. https://doi.org/10.1109/ICCCNT45670.2019.8944432.
[39] S. Haman et al., "Energy consumption reduction between connected objects in MQTT-based networks," J. Comput. Commun., vol. 12, no. 10, pp. 177–188, 2024. https://doi.org/10.4236/jcc.2024.1210012.
[40] D. Z. Fawwaz et al., "Optimal distributed MQTT broker and services placement for SDN-edge smart city architecture," Sensors, vol. 22, no. 9, p. 3431, Apr. 2022. https://doi.org/10.3390/s22093431.
[41] K. Kosaka, Y. Noda, T. Yokotani, and K. Ishibashi, "Implementation and evaluation of the control mechanism among distributed MQTT brokers," IEEE Access, vol. 11, pp. 134211–134216, 2023. https://doi.org/10.1109/ACCESS.2023.3335273.
[42] A. F. Oklilas, R. Zulfahmi, Ermatita, and A. P. Jaya, "Temperature monitoring system based on Message Queue Telemetry Transport (MQTT)," in Proc. Int. Conf. Informatics, Multimedia, Cyber Inf. Syst. (ICIMCIS), Jakarta, Indonesia, Oct. 2019, pp. 61–66. https://doi.org/10.1109/ICIMCIS48181.2019.8985356.
[43] R. Kawaguchi and M. Bandai, "Edge-based MQTT broker architecture for geographical IoT applications," in Proc. Int. Conf. Inf. Netw. (ICOIN), Barcelona, Spain, Jan. 2020, pp. 232–235. https://doi.org/10.1109/ICOIN48656.2020.9016528.
[44] E. Shahri, P. Pedreiras, and L. Almeida, "A scalable real-time SDN-based MQTT framework for industrial applications," IEEE Open J. Ind. Electron. Soc., vol. 5, pp. 215–235, 2024. https://doi.org/10.1109/OJIES.2024.3373232.
[45] K. Sahlmann, V. Clemens, M. Nowak, and B. Schnor, "MUP: Simplifying secure over-the-air update with MQTT for constrained IoT devices," Sensors, vol. 21, no. 1, p. 10, Dec. 2020. https://doi.org/10.3390/s21010010.
[46] F. Buccafurri and C. Romolo, "A blockchain-based OTP authentication scheme for constrained IoT devices using MQTT," in Proc. Int. Symp. Comput. Sci. Intell. Control, Amsterdam, Netherlands, Sep. 2019, pp. 1–5. https://doi.org/10.1145/3386164.3389095.
[47] W.-T. Su, W.-C. Chen, and C.-C. Chen, "An extensible and transparent thing-to-thing security enhancement for MQTT protocol in IoT environment," in Proc. Global IoT Summit (GIoTS), Aarhus, Denmark, Jun. 2019, pp. 1–4. https://doi.org/10.1109/GIOTS.2019.8766412.
[48] B. Khan and C. Pirak, "Experimental performance analysis of MQTT and CoAP protocol usage for NB-IoT smart meter," in Proc. Int. Electr. Eng. Congr. (iEECON), Pattaya, Thailand, Mar. 2021, pp. 65–68. https://doi.org/10.1109/iEECON51072.2021.9440273.
[49] W. Yan et al., "Survey on recent smart gateways for smart home: Systems, technologies, and challenges," Trans. Emerg. Telecommun. Technol., vol. 33, no. 6, p. e4067, Jun. 2022. https://doi.org/10.1002/ett.4067.
[50] A. Zainudin, M. F. Syaifudin, and N. Syahroni, "Design and implementation of node gateway with MQTT and CoAP protocol for IoT applications," in Proc. Int. Conf. Inf. Technol., Inf. Syst. Electr. Eng. (ICITISEE), Yogyakarta, Indonesia, Nov. 2019, pp. 155–159. https://doi.org/10.1109/ICITISEE48480.2019.9003734.
[51] N. Nikolov, "Research of MQTT, CoAP, HTTP and XMPP IoT communication protocols for embedded systems," in Proc. Int. Sci. Conf. Electronics (ET), Sozopol, Bulgaria, Sep. 2020, pp. 1–4. https://doi.org/10.1109/ET50336.2020.9238208.
[52] N. K. Ribas and M. A. Spohn, "A new approach to a self-organizing federation of MQTT brokers," J. Comput. Sci., vol. 18, no. 7, pp. 687–694, Jul. 2022. https://doi.org/10.3844/jcssp.2022.687.694.
[53] T. Zeybek, C. H. Chang, and Z. Yang, "An IoT implementation for manufacturing using Wi-Fi, 6LoWPAN, and MQTT," ACM, 2018. https://doi.org/10.5555/3324320.3324410.
[54] I. K. A. A. Aryanto et al., "Design of soil humidity monitoring system using the Internet of Things concept and MQTT," in Proc. Int. Conf. Smart Technol. Appl. (ICoSTA), Surabaya, Indonesia, Feb. 2020, pp. 1–6. https://doi.org/10.1109/ICoSTA48221.2020.1570611115.
[55] A. K. Saputro et al., "Application of LoRa in optimizing Internet of Things using MQTT for fish feed monitoring," in Proc. Inf. Technol. Int. Seminar (ITIS), Surabaya, Indonesia, Oct. 2020, pp. 224–228. https://doi.org/10.1109/ITIS50118.2020.9321021.
[56] A. Huang et al., "A practical marine wireless sensor network monitoring system based on LoRa and MQTT," in Proc. IEEE Int. Conf. Electron. Technol. (ICET), Chengdu, China, May 2019, pp. 330–334. https://doi.org/10.1109/ELTECH.2019.8839464.
[57] E. Longo and A. E. C. Redondi, "Design and implementation of an advanced MQTT broker for distributed pub/sub scenarios," Comput. Netw., vol. 224, p. 109601, Apr. 2023. https://doi.org/10.1016/j.comnet.2023.109601.
[58] M. Kashyap, A. K. Dev, and V. Sharma, "Implementation and analysis of EMQX broker for MQTT protocol in the Internet of Things," E-Prime – Adv. Electr. Eng. Electron. Energy, vol. 10, p. 100846, Dec. 2024. https://doi.org/10.1016/j.prime.2024.100846.
[59] F. Pazos, "Evaluación de desempeño de servidores supervisores MQTT instalados en la nube," SADIO Electron. J. Inform. Oper. Res., vol. 23, no. 1, p. e043, Apr. 2024. https://doi.org/10.24215/15146774e043.
[60] D. Borsatti et al., "From IoT to cloud: Applications and performance of the MQTT protocol," in Proc. Int. Conf. Transparent Opt. Netw. (ICTON), Bari, Italy, Jul. 2020, pp. 1–4. https://doi.org/10.1109/ICTON51198.2020.9203167.
[61] X. You et al., "Epsilon-greedy-based MQTT QoS mode selection and power control algorithm for power distribution IoT," Int. J. Mob. Comput. Multimed. Commun., vol. 14, no. 1, pp. 1–18, Mar. 2023. https://doi.org/10.4018/IJMCMC.306976.
[62] I. Vaccari et al., "MQTTset: A new dataset for machine learning techniques on MQTT," Sensors, vol. 20, no. 22, p. 6578, Nov. 2020. https://doi.org/10.3390/s20226578.
[63] A. S. Sadeq et al., "A QoS approach for Internet of Things environment using MQTT protocol," in Proc. Int. Conf. Cybersecurity (ICoCSec), Negeri Sembilan, Malaysia, Sep. 2019, pp. 59–63. https://doi.org/10.1109/ICoCSec47621.2019.8971097.
[64] A. Shalaginov, O. Semeniuta, and M. Alazab, "MEML: Resource-aware MQTT-based machine learning for network attacks detection on IoT edge devices," in Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion, Auckland, New Zealand, Dec. 2019, pp. 123–128. https://doi.org/10.1145/3368235.3368876.
[65] G. Wang, "The study of IoT MQTT publish latency," in Proc. IEEE Int. Conf. Electro Inf. Technol. (EIT), Mt. Pleasant, MI, USA, May 2021, pp. 27–31. https://doi.org/10.1109/EIT51626.2021.9491838.
[66] M. Dave, J. Doshi, and H. Arolkar, "MQTT–CoAP interconnector: IoT interoperability solution for application layer protocols," in Proc. Int. Conf. I-SMAC, Palladam, India, Oct. 2020, pp. 122–127. https://doi.org/10.1109/I-SMAC49090.2020.9243377.
[67] L. Raju K. and V. Vijayaraghavan, "IoT- and cloud-hinged smart irrigation system employing MQTT protocol," in Proc. Int. Conf. Devices, Circuits Syst. (ICDCS), Coimbatore, India, Mar. 2020, pp. 71–75. https://doi.org/10.1109/ICDCS48716.2020.243551.
[68] V. Tejashree et al., "MQTT-SN based architecture for estimating delay and throughput in IoT," in Data Science and Computational Intelligence, vol. 1483, Cham, Switzerland: Springer, 2021, pp. 481–490. https://doi.org/10.1007/978-3-030-91244-4_38.
[69] A. Velinov et al., "Covert channels in the MQTT-based Internet of Things," IEEE Access, vol. 7, pp. 161899–161915, 2019.

https://doi.org/10.1109/ACCESS.2019.2951425.
[70] M. H. Widianto et al., "Energy saving on IoT using LoRa: A systematic literature review," Int. J. Reconfigurable Embed. Syst., vol. 11, no. 1, p. 25, Mar. 2022. https://doi.org/10.11591/ijres.v11.i1.pp25-33.
[71] M. A. Khan et al., "A deep learning-based intrusion detection system for MQTT-enabled IoT," Sensors, vol. 21, no. 21, p. 7016, Oct. 2021. https://doi.org/10.3390/s21217016.
[72] D. B. C. Lima et al., "A performance evaluation of Raspberry Pi Zero W based gateway running MQTT broker for IoT," in Proc. IEEE IEMCON, Vancouver, BC, Canada, Oct. 2019, pp. 76–81. https://doi.org/10.1109/IEMCON.2019.8936206.
[73] A. Thantharate, C. Beard, and P. Kankariya, "CoAP- and MQTT-based models for OTA updates to IoT devices," in Proc. IEEE iThings/GreenCom/CPSCom/SmartData, Atlanta, GA, USA, Jul. 2019, pp. 1065–1070. https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00183.
[74] P. S. Akshatha and S. M. Dilip Kumar, "MQTT and blockchain sharding: An approach to user-controlled data access," Blockchain Res. Appl., vol. 4, no. 4, p. 100158, Dec. 2023. https://doi.org/10.1016/j.bcra.2023.100158.
[75] F. Buccafurri, V. De Angelis, and R. Nardone, "Securing MQTT by blockchain-based OTP authentication," Sensors, vol. 20, no. 7, p. 2002, Apr. 2020. https://doi.org/10.3390/s20072002.
[76] M. A. Bin Ahmadon, N. Yamaguchi, and S. Yamaguchi, "Process-based intrusion detection method for IoT system with MQTT protocol," in Proc. IEEE GCCE, Osaka, Japan, Oct. 2019, pp. 953–956. https://doi.org/10.1109/GCCE46687.2019.9015252.
[77] A. Munshi, "Improved MQTT secure transmission flags in smart homes," Sensors, vol. 22, no. 6, p. 2174, Mar. 2022. https://doi.org/10.3390/s22062174.
[78] M. I. Younas et al., "Toward QoS monitoring in IoT edge devices driven healthcare: A systematic literature review," Sensors, vol. 23, no. 21, p. 8885, Nov. 2023. https://doi.org/10.3390/s23218885.
[79] A. J. Hintaw et al., "A robust security scheme based on enhanced symmetric algorithm for MQTT in the Internet of Things," IEEE Access, vol. 11, pp. 43019–43040, 2023. https://doi.org/10.1109/ACCESS.2023.3267718.
[80] C. R. M. Silva and F. A. C. M. Silva, "An IoT gateway for Modbus and MQTT integration," in Proc. SBMO/IEEE MTT-S IMOC, Aveiro, Portugal, Nov. 2019, pp. 1–3. https://doi.org/10.1109/IMOC43827.2019.9317637.
[81] N. K. Bharti et al., "Hydroponics system integrated with Android application using IoT and MQTT broker," in Proc. IEEE PuneCon, Pune, India, Dec. 2019, pp. 1–5. https://doi.org/10.1109/PuneCon46936.2019.9105847.
[82] F. Masykur et al., "Application of MQTT protocol in IoT to monitor mushroom cultivation," in Proc. Int. Conf. IT, Comput. Electr. Eng. (ICITACEE), Semarang, Indonesia, Sep. 2020, pp. 135–139. https://doi.org/10.1109/ICITACEE50144.2020.9239118.
[83] E. Ciklabakkal et al., "ARTEMIS: An intrusion detection system for MQTT attacks in IoT," in Proc. IEEE Symp. Reliable Distrib. Syst. (SRDS), Lyon, France, Oct. 2019, pp. 369–372. https://doi.org/10.1109/SRDS47363.2019.00053.
[84] M. Michaelides, C. Sengul, and P. Patras, "An experimental evaluation of MQTT authentication and authorization in IoT," in Proc. ACM Workshop Wireless Netw. Testbeds Exp. Eval. Characterization, New Orleans, LA, USA, Jan. 2022, pp. 69–76. https://doi.org/10.1145/3477086.3480838.
[85] M. Markovic and P. Edwards, "Enhancing transparency of MQTT brokers for IoT applications through provenance streams," in Proc. Int. Workshop Middleware Appl. IoT, Davis, CA, USA, Dec. 2019, pp. 17–20. https://doi.org/10.1145/3366610.3368099.