MQTT in Focus: Understanding the Protocol and Its Recent Advancements

Nael M. Radwan *nradwan@uidaho.edu*
Computer Science Department
Center for Secure & Dependable Systems (CSDS)
University of Idaho
Moscow, Idaho,83843, USA

Jim Alves-Foss *jimaf@uidaho.edu*
Computer Science Department
Director of Center for Secure & Dependable Systems (CSDS)
University of Idaho
Moscow, Idaho,83843, USA

Abstract

This article offers a detailed exploration of MQTT (Message Queuing Telemetry Transport) and its latest version, MQTTv5. It delves into MQTT's components, protocol layers, and the pivotal role of brokers in MQTT networks. We differentiate between public and private brokers, outlining their use cases. The article focuses on MQTT's flow control mechanisms, emphasizing MQTTv5's enhancements with sending quotas and the Receive Maximum attribute. It discusses limitations related to message Quality of Service (QoS) levels. We also address MQTT's challenges, including TCP reliance, scalability issues, single points of failure, implementation complexity, and TCP/IP support requirements. This information equips readers with insights to make informed decisions for IoT projects.

Keywords: The Internet of Things (IoT), Message Queuing Telemetry Transfer (MQTT), Messaging Protocols (MP).

1. INTRODUCTION

The Internet of Things (IoT) ecosystem serves as a dynamic platform enabling uniquely identifiable devices to connect to the Internet. This connectivity empowers these devices to seamlessly share information with humans. This ecosystem continues to expand rapidly, facilitating global intercommunication among an extensive array of devices(Radwan 2020). The driving force behind this remarkable growth is the ever-increasing demand for convenient access to data to enhance services. This surge has resulted in ubiquitous connectivity, uniting various devices within smart grids, health monitoring systems, home networks, building automation, vehicular communication, and telecommunication networks. As a result, a diverse range of devices, including health monitors, sensors, industrial automation tools, vehicles, and home appliances, now boast internet connectivity. This proliferation has ushered in a new era of IoT (Al-Hawawreh and Sitnikova 2020).

The primary objective of the IoT is to establish a connected ecosystem where devices can seamlessly communicate with each other over the Internet. Achieving this objective necessitates efficient inter-operation among the numerous Device-to-Device (D2D) communication technologies that compose the IoT ecosystem. Currently, these technologies operate within vertical silos, each using distinct protocols. To tackle this challenge, this study delves into the complexities associated with integrating and achieving interoperability among D2D technologies, with a specific emphasis on network layer functions such as addressing, routing, mobility, security, and resource optimization(Al-Hawawreh and Sitnikova 2020, Radwan 2020).

As IoT devices proliferate at an unprecedented rate, experts estimate that by 2025, there will be over 20 billion connected devices. With IoT devices increasingly becoming an integral part of our daily lives, security has become a paramount concern within the IoT ecosystem. The IoT has evolved into a next-generation IT platform, fostering connectivity between physical devices and a diverse spectrum of services via the Internet. This transformation has been made possible through a set of protocols designed for data collection and transfers over the internet, harnessing wireless communication technologies like Wi-Fi and ZigBee. The growing number of smart devices has led to the creation of a distributed database within a network of interconnected devices(Ray 2017).

The characteristics of IoT encompass connectivity, smart sensing, intelligence, energy efficiency, and safety, all aimed at enhancing various aspects of human life. These characteristics give rise to an array of applications, including intelligent analytics, improved security, enhanced productivity, efficient inventory management, secure travel, and real-time data processing(Ray 2017).

As the number of smart devices, wireless technologies, and sensors continues to proliferate, there is an escalating need to comprehensively study the infrastructure of the Internet of Things. The structural elements of the Internet of Things are classified based on various indicators, applications, technologies, works, objectives, architectural requirements, network topologies, and the inherent diversity of the IoT platform's architecture(Vongsingthong and Smanchat 2014). Key applications of the Internet of Things include smart transportation, smart homes, smart healthcare, smart grids, intelligent lighting, and automated buildings, all of which significantly enrich people's daily lives.

IoT is often categorized into three paradigms: internet-oriented, sensors, and knowledge. The implementation of Internet of Things technology is typically near modern society, as wireless sensor systems inherently bridge the gap between people and things(Radwan 2020).

In the distributed environment known as the Fog, situated in an intermediate architectural layer adjacent to the network edge and user devices(Rahmani, Gia et al. 2018), the Internet of Things offers myriad opportunities for the connected healthcare industry. Fog computing is expected to deliver various advantages, including reduced latency, local control, enhanced security, increased reliability, fault tolerance, lower data transfer and storage costs, extensibility, distributed computing, effective heterogeneity management, and support for hardware and software maintenance(Kumari, Tanwar et al. 2018).

The emergence of the Internet of Things has enabled the integration of physical objects, sensors, and computational elements into a seamlessly connected environment. This integration gives rise to smart environments that continually interact with their inhabitants, to enhance and support human abilities. For example, these environments can assist the elderly in navigating unfamiliar spaces or in tasks such as moving heavy objects. Researchers have made numerous attempts to harness the potential of the Internet of Things to simplify our daily lives and explore the impact of IoT-based smart environments on human life(Kumari, Tanwar et al. 2018, Rahmani, Gia et al. 2018).

Emerging technologies such as IoT and Software-Defined Networks (SDN) aim to connect objects over the Internet and provide orchestration for network management, respectively. With billions of connected objects, effectively managing and controlling them over large distribution networks presents a complex challenge. Although various solutions have been explored to address the challenges within the Internet of Things paradigm, traditional networks are ill-equipped to handle the enormous number of connected devices and the associated data manipulation(Kumari, Tanwar et al. 2018, Wang, Ji et al. 2021).

According to Ray(2017), Software-Defined Networking (SDN) stands as a revolutionary technology that supports heterogeneous networking through programmable planes, offering rapid

evolution and dynamism. The integration of SDN and the Internet of Things holds the potential to meet control and management expectations in a variety of scenarios. However, as highlighted in, the current Internet of Things faces a slew of security issues, including vulnerabilities in IoT systems, malware detection, data security concerns, personal and public safety risks, privacy issues, and the management of data storage following the exponential growth of IoT devices(Ray 2017, Al-Hawawreh and Sitnikova 2020).

Today, many home devices are connected to the internet through messaging protocols (MP), offering customers increased convenience and accessibility. Nevertheless, most MPs within Internet of Things platforms remain fragmented and lack the necessary security measures(Wang, Ji et al. 2021). In response to this challenge, MP-Inspector has been developed as an automatic and systematic solution to verify the security of MP implementations. MP-Inspector leverages model learning and formal analysis across three stages: (a) automatic inference of the state machine of an MP implementation through parameter semantics extraction and interaction logic extraction, (b) generation of security properties based on meta properties and the state machine, and (c) the application of automatic property-based formal verification to identify property violations (Wang, Ji et al. 2021). By utilizing MP-Inspector, it becomes possible to ensure that both convenience and security are upheld when utilizing messaging protocols on Internet of Things devices(Vongsingthong and Smanchat 2014, Kumari, Tanwar et al. 2018, Rahmani, Gia et al. 2018).

However, as indicated by Wang et al(2021), there remains a lack of a comprehensive and automated approach to verify the security of messaging protocols (MP) due to the diversity and customization of MP implementations, as well as the closed-source nature of many MP workflows. To tackle these challenges, the proposed solution, MP-Inspector, stands as the first systematic and automatic approach for detecting security vulnerabilities in MP implementations. This framework adopts a property-driven and model-based testing approach, involving the modeling of an MP implementation into a state machine, the extraction of security properties from the standard MP specification, and the refinement of these properties based on the learned state machine. Finally, the state machine is analyzed through formal verification to detect any property violations (Obaidat, Obeidat et al. 2020).

The IoT is structured around three essential layers: the perception layer, the network layer, and the application layer. While the fundamental security objectives of Confidentiality, Integrity, and Availability (CIA) apply to the Internet of Things, there are additional security concerns. These include general security features and specific security issues. These security principles are paramount in establishing a secure communication framework among people, programs, processes, and things. Among these principles are confidentiality, integrity, availability, and authentication, as well as the implementation of encryption and authentication protocols for data and devices within IoT(Obaidat, Obeidat et al. 2020). Additional principles encompass heterogeneity between devices, policies for data management and protection, and key management systems for the encryption process, ultimately ensuring the confidentiality of data(Lara, Aguilar et al. 2020).

2. MQTT PROTOCOL

Message Queuing Telemetry Transfer (MQTT) is a widely used publish/subscribe protocol in the realm of the Internet of Things (IoT). It was developed in 1999 by IBM as a lightweight messaging protocol designed for devices operating in high-disconnection environments. MQTT is based on the TCP protocol(Ray 2017, Kumari, Tanwar et al. 2018, Obaidat, Obeidat et al. 2020, Radwan 2020).

MQTT is a widely used communication protocol on the Internet of Things. According to Liu et al.(2020), it excels at exchanging information over long distances, especially for transmitting small data between devices and software/SCADA systems. SCADA stands for Supervisory Control and Data Acquisition, which manages machinery and processes through computer networks. MQTT's

back-filling capability simplifies network configuration, making it apt for low-frequency, low-bandwidth networks. There have been efforts to enhance MQTT security, including a proprietary learning-based obfuscation method evaluated in research (Liu, Zhang et al. 2020).

2.1 Work of the MQTT Protocol

MQTT operates on a publish/subscribe model. Clients can either subscribe to specific topics to receive messages or publish messages on topics. The central component of MQTT is the MQTT broker, which handles client requests. When a client publishes a message on a topic, the broker forwards this message to all other clients subscribed to that topic. MQTT's reliability is highly dependent on the broker's functionality, which makes it a target for improving network security(Bender, Kirdan et al. 2021).

2.2 MQTT Benefits

There are several reasons to use MQTT:

- It's open-source, simple, and lightweight.
- Ideal for transferring small amounts of data between devices, as well as between devices and software/SCADA systems(Liu, Zhang et al. 2020).
- Suitable for wireless networks with low power consumption.
- Fast data transfer, allowing for high reliability when needed.
- It's resource-efficient, requiring minimal CPU and memory usage, making it well-suited for physical computing(Froiz-Míguez, Fernández-Caramés et al. 2018).

2.3 How the Protocol Works (Principle of Use)

MQTT differs from traditional internet communication methods like HTTP. It functions on a system where all users are connected to a medium for information exchange. There are two main categories of users: publishers and subscribers. Information is published on specific channels called "topics," and users subscribe to these topics to receive the data they require. Topics are hierarchical, enabling users to select specific information (Hunkeler, Truong and Stanford-Clark 2008, Bhawiyuga, Data and Warda 2017).

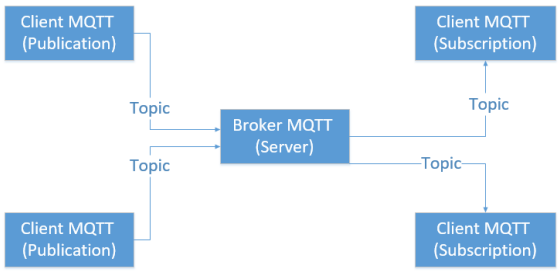

FIGURE 1: MQTT Broker.

For example, if you have sensors in a house, each sensor publishes data to the broker based on its location, e.g., "home/sensor/living-room." A sensor doesn't send data directly to the device requesting it; instead, the broker locates the requesting device and delivers the data (Froiz-Míguez, Fernández-Caramés et al. 2018). MQTT allows securing communication through various methods such as SSL/TLS, certificate-based authentication, or username and password authentication (Hunkeler, Truong and Stanford-Clark 2008).

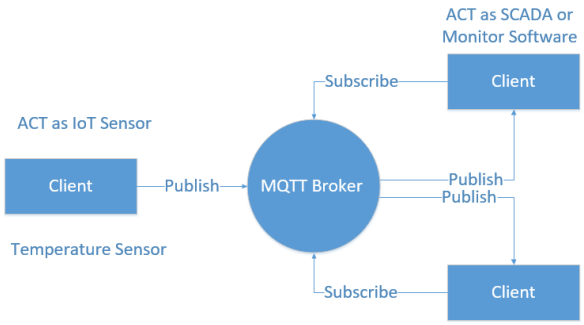

FIGURE 2: MQTT Work: schematic data flow from the sensor (machine) to devise (machine).

2.4 Protocol Publisher/Subscriber Logic

Publishers own the information and initiate communication with the broker via a unique ID. They send information at specified intervals and log their actions. Subscribers, on the other hand, initiate communication, listen for incoming information, and respond to it if it's what they need. They also log their actions for troubleshooting purposes.

2.5 Data Integrity in the MQTT Protocol (Quality of Service)

According to Gupta, data integrity in MQTT is upheld through its Quality of Service (QoS) levels. There are three QoS levels:

1. **Level 0-** "at most once." Messages are sent with no guarantee of arrival. This is like a "fire and forget" approach, where the sender doesn't require confirmation of message delivery (Bender, Kirdan et al. 2021).
2. **Level 1-** "at least once." Messages are sent multiple times, if necessary, until the broker confirms delivery. The sender retains the message until it receives a Publish-Back (packet ID) from the recipient, ensuring at least one successful delivery (Froiz-Míguez, Fernández-Caramés et al. 2018).
3. **Level 2-** "exactly once." Messages are saved until the subscriber acknowledges receipt. This level ensures the highest quality but is slower and more resource intensive. It employs a four-part handshake to guarantee exclusive message delivery (Hunkeler, Truong and Stanford-Clark 2008).

Here's how it works in more detail:

- **LEVEL 0:** Level 0 doesn't guarantee message delivery. (Bender, Kirdan et al. 2021), as in Figure 3.

FIGURE 3: Quality of Service level 0: delivery at most once.

- **LEVEL 1:** Ensure messages are delivered at least once, with retransmission if necessary. (Froiz-Míguez, Fernández-Caramés et al. 2018), as in Figure 4.

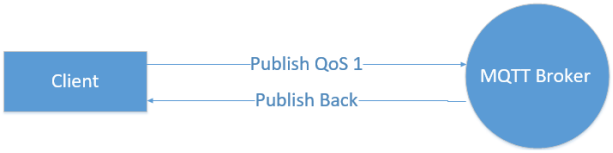

FIGURE 4: Quality of Service level 1: delivery at least once.

- **LEVEL 2:** guarantees that each message is delivered exactly once but is the slowest option and involves a detailed confirmation process.

To achieve Level 2, a sender and recipient go through a series of acknowledgments. The sender initially sends a Publish packet, and the recipient responds with a Publish-Received packet. If the sender doesn't receive this acknowledgment, it resends the Publish packet. Once acknowledged, the sender sends a Publish-Release packet, and the recipient replies with a Publish-Complete packet. This process ensures that the message is delivered and received with confirmation. If a packet is lost, the sender is responsible for retransmitting it. Both MQTT clients and brokers follow this protocol to ensure message integrity(Andy, Rahardjo and Hanindhito 2017, Bhawiyuga, Data and Warda 2017, Gupta, Khera and Turk 2021, Kumar, Sharma et al. 2021), as in Figure 5.

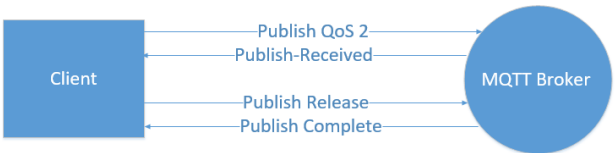

FIGURE 5: Quality of Service level 2: delivery exactly once.

2.6 MQTT Architecture

MQTT architecture consists of two primary components:

a) **Client:** Establishes network connections to the MQTT broker and can function as both a publisher and subscriber. It can publish messages, subscribe to topics, unsubscribe from topics, and disconnect from the broker (Soni and Makwana 2017, Yassein, Shatnawi et al. 2017).

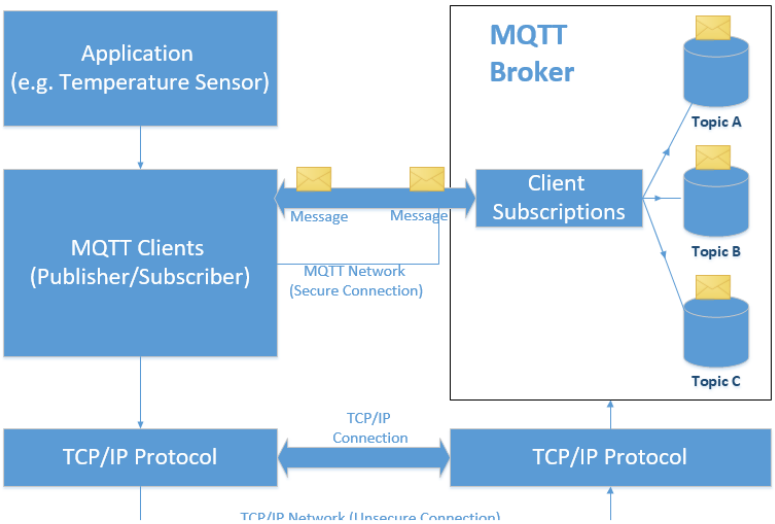

FIGURE 6: MQTT Architecture.

b) **Broker:** Controls the distribution of information, receiving messages from publishers, filtering them, determining recipients, and forwarding messages to subscribers. The broker also handles client requests, including subscriptions and unsubscriptions. This architecture enables efficient communication between devices and is integral to the operation of MQTT in IoT scenarios. These detailed points provide a comprehensive understanding of MQTT and its role in IoT applications (Soni and Makwana 2017, Yassein, Shatnawi et al. 2017).

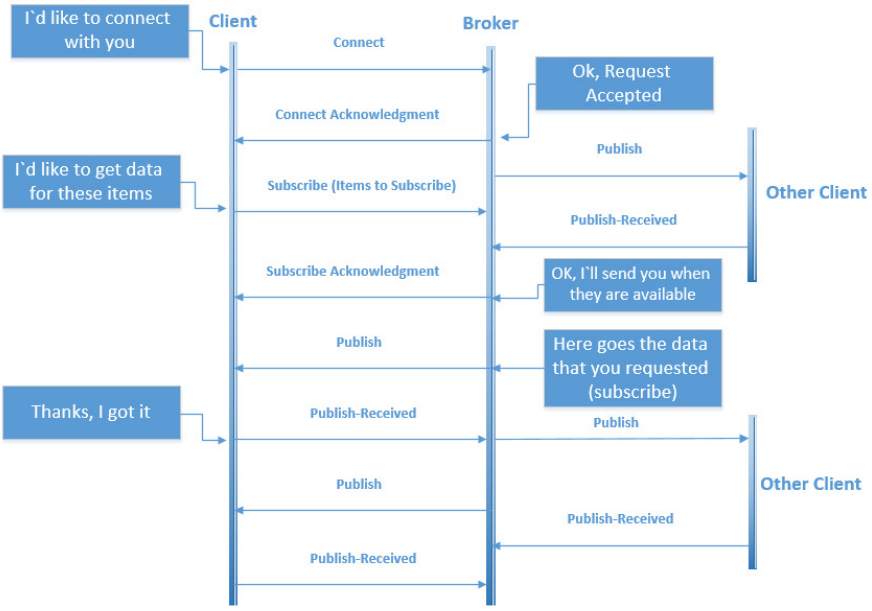

FIGURE 7: Working of MQTT.

3. BROKERS

In MQTT, the central element is the broker, a crucial component for establishing MQTT networks. Brokers can be hosted online in the cloud or locally on a private device (Dinculeană and Cheng 2019). Many brokers are open source, allowing users to obtain credentials, install, and set them up on their devices. This sets up a broker with a specific IP, making it part of the private local network, akin to a SCADA server, where clients interact with the server(Grgić, Špeh and Heđi 2016, Atmoko, Riantini and Hasin 2017).

3.1 Types of Brokers

Every broker focuses on essential components, with the main ones being the broker's address (IP Broker), the TCP Port, and the TLS Port, ensuring secure communication between clients and the server or between clients. The MQTT protocol typically uses Port 1883 by default, with other ports considered custom and can be set as needed(Hiromoto, Haney and Vakanski 2017).

For TCP communication, the critical factors are the IP and port. A specific port is essential for TCP communication to occur. This approach organizes services within defined ports, enhancing communication efficiency. For instance, the HTTP protocol relies on Port 80 for TCP-based communication, although other ports can be used for similar communication types(Hiromoto, Haney and Vakanski 2017).

This design choice ensures that data shared between clients and the broker remains encrypted and secure. Users on the same network cannot decrypt the data, and only the intended recipients can access it. TLS connections use assigned specific ports, while TCP connections also require specific ports. Just like HTTP and HTTPS have distinct ports, a TLS connection is encrypted, securing data transmission, and protecting sensitive information like usernames and passwords (Hiromoto, Haney and Vakanski 2017).

Another method to view the connection on the broker is through a web socket, although it doesn't employ the MQTT protocol. Instead, it uses a different protocol called Web-socket. This method provides insights into the server's functioning, monitoring messages sent and received, and tracking delivery attempts(Karagiannis, Chatzimisios et al. 2015).

3.2 Brokers (Public/Private)

MQTT brokers come in two types: public and private. Public brokers allow any device to publish and subscribe to topics without privacy restrictions(Ferrari, Flammini et al. 2018). However, they are not recommended for production use and are better suited for learning or experimentation. In contrast, private brokers are more secure. Only devices authorized by the user can publish and subscribe to the broker's topics. Private brokers are suitable for both production and prototyping purposes(Atmoko, Riantini and Hasin 2017).

In summary, brokers are central to MQTT networks, enabling communication between devices. Understanding broker types and their associated security measures is crucial for configuring effective MQTT networks. (See Appendix 1).

4. MQTT PROBLEMS

According to Ferrari, to attain a deeper understanding of MQTT's security, it is essential to consider security from the network layer to the application layer. Notably, selecting an MQTT broker designed for security is crucial. While open-source brokers may suffice for home automation, professional deployments demand heightened security considerations (Swamy, Jadhav and Kulkarni 2017, Ferrari, Flammini et al. 2018).

4.1 Problem Statement

MQTT plays a pivotal role in critical global deployments for device connectivity. Security is a paramount concern for such deployments. MQTT, being a layer 7 protocol, relies on underlying layers, and it is typically used in conjunction with TLS to establish encrypted communication channels between devices and brokers (Swamy, Jadhav and Kulkarni 2017, Ferrari, Flammini et al. 2018).

Despite these security measures, MQTT deployments invariably emphasize user authentication and authorization mechanisms. These mechanisms ensure that only authorized devices can connect to the broker. Devices must present valid credentials, certificates, and authorization to publish and subscribe to specific topics and perform designated actions. Nevertheless, MQTT's security is not without its vulnerabilities, and there is evidence to support this claim (Upadhyay, Borole and Dileepan 2016).

4.2 Problem Analysis

Despite MQTT's numerous advantages, it may not be suitable for all scenarios, as observed in (Upadhyay, Borole and Dileepan 2016, Swamy, Jadhav and Kulkarni 2017, Ferrari, Flammini et al. 2018). Some of the protocol's drawbacks include:

1. Slower transmit cycles when compared to the Constrained Application Protocol (CoAP), which can be critical for systems with a substantial number of devices (Ferrari, Flammini et al. 2018).
2. Challenges in resource discovery, given that MQTT employs a flexible topic subscription system, whereas CoAP employs a stable resource discovery system (Ferrari, Flammini et al. 2018).
3. Limited security encryption. MQTT primarily lacks encryption, despite TLS/SSL usage (Swamy, Jadhav and Kulkarni 2017).
4. Scalability concerns, particularly in establishing globally scalable networks when compared to alternative protocols (Upadhyay, Borole and Dileepan 2016).

While MQTT remains a fitting solution for many applications, addressing its security and interoperability challenges necessitates consultation with Internet of Things development experts (Swamy, Jadhav and Kulkarni 2017).

4.3 Problem Explanation

MQTT communication may become unreliable, and a publisher might disconnect without warning. In anticipation of such situations, a publisher can set up a "last will and testament." This message

is stored on the broker and is dispatched to subscribers in the event of an unanticipated disconnection. The message typically contains information to identify the disconnected publisher and guide appropriate actions (Upadhyay, Borole and Dileepan 2016).

MQTT was initially designed to facilitate efficient data transmission over unreliable and costly communication lines. As a result, security received limited attention during its design and implementation. Nevertheless, some security options exist, albeit with increased data transmission and storage requirements. Network security can be an effective measure, where the network itself is secured, making the transmission of unencrypted MQTT data irrelevant. Notably, security breaches must arise from within the network, such as through malicious actors or network intrusions (Upadhyay, Borole and Dileepan 2016).

MQTT allows the use of usernames and passwords for establishing connections with brokers. Regrettably, in pursuit of lightweight communication, these usernames and passwords are transmitted in plain text. This approach was acceptable in 1999 but is insecure in today's context, where wireless network communications can be easily intercepted. While usernames and passwords can prevent unintentional connections, they offer little protection against malicious actors. The commonly employed SSL/TLS, which runs atop TCP/IP, introduces significant overhead to the otherwise lightweight MQTT communication (Swamy, Jadhav and Kulkarni 2017).

MQTT communication between the client and broker involves the generation of TCP/IP packets by the client, which are processed to generate specific outputs sent to the broker. The broker, upon receiving these packets, responds with TCP/IP packets, and the client deciphers the resulting output and forwards it to the mapper for abstraction. However, MQTT brokers may exhibit non-deterministic behavior due to factors such as latency or timeouts (Upadhyay, Borole and Dileepan 2016, Swamy, Jadhav and Kulkarni 2017).

4.4 Problem Conclusion

One primary source of security and privacy issues on the Internet of Things is the prevalence of insecure default configurations. A default misconfigured MQTT server poses significant risks, as anyone with access can intercept all messages transmitted through it due to MQTT's use of wildcards. Alarmingly, many MQTT servers are poorly configured and accessible without authentication on the public internet. For instance, the world's first Internet of Things search engine uncovered almost 67,000 exposed MQTT servers, most of which lack authentication. The MQTT protocol allows anyone to subscribe to broadcasted topics without authentication, making it highly vulnerable. Fundamental security principles for the Internet of Things include data confidentiality, availability, integrity, and privacy (Upadhyay, Borole and Dileepan 2016).

The current implementation of MQTT primarily provides identity, authentication, and authorization within its security mechanisms. However, relying on TCP/IP for sending/receiving data is insecure, even though TCP/IP packets are generated and sent to the broker. MQTT clients can subscribe to topic filters, publish messages to topic names, and receive messages from subscribed topics. Notably, the topic filter allows the use of wildcard characters for simultaneous subscriptions or subscriptions to multiple topics (Upadhyay, Borole and Dileepan 2016, Swamy, Jadhav and Kulkarni 2017).

Despite MQTT's widespread use, it was not originally designed with a strong emphasis on security, as evidenced by its standing as the second most popular Internet of Things messaging protocol. The substantial growth of MQTT's usage demands the development of a secure version that accounts for resource-constrained devices. Therefore, prioritizing MQTT security requirements is essential (Andy, Rahardjo and Hanindhito 2017).

This section presents diverse security implementations for MQTT in real-world applications. For devices without constraints, TLS/SSL can be used to establish transport-level encryption, as proposed by (Soni and Makwana 2017). According to (Dinculeană and Cheng 2019), introduced

an alternative approach to end-to-end security and hop-by-hop protection through the link layer, successfully implemented in an industrial wind park(Dinculeană and Cheng 2019). Additionally, Soni offered a solution for enforcing security policy rules at the MQTT layer, utilizing Model-based Security.

5. RELATED WORK

5.1 Flow Control

MQTT servers often have fixed and limited resources, while clients' flow can fluctuate unpredictably. To ensure system stability, flow control mechanisms are necessary. Common flow control algorithms include the sliding window counting method, leaky bucket algorithm, and token bucket algorithm.

While MQTT v3 did not standardize flow control behavior, MQTT v5 introduced flow control functionality, which will be discussed further (Bashir and Mir 2020, Fauzan, Sukarno and Wardana 2020).

5.2 Flow Control in MQTT V5

In MQTT v5, the sender initially has a sending quota, and this quota decreases by one whenever a PUBLISH packet with a QoS greater than 0 is sent. Upon receiving response packets (PUBACK, PUBCOMP, or PUBREC), the sending quota is increased by one.

If the receiver does not respond immediately, the sending quota is reduced to 0, and the sender must stop sending all PUBLISH packets with QoS greater than 0 until the quota recovers. This algorithm resembles a variation of the token bucket algorithm. It effectively utilizes resources by not limiting the receiving rate, with the sending rate dependent on the response rate of the opposite end and network conditions (Bashir and Mir 2020, Fauzan, Sukarno and Wardana 2020).

5.3 Get Maximum Attribute

MQTT v5 introduced the Receive Maximum attribute to facilitate flow control (Fauzan, Sukarno and Wardana 2020). It appears in the CONNECT and CONNACK packets and represents the maximum number of PUBLISH packets with a QoS of 1 and 2 that the client and server can handle simultaneously.

This attribute determines the maximum sending quota the opposite end can utilize (Fauzan, Sukarno and Wardana 2020). While MQTT allows messages up to 256 MB in size, it is common for messages to be smaller than 10 KB due to network constraints and the inability to retry sending large messages (Frustaci, Pace et al. 2017, Qiu, Tian et al. 2020).

5.4 Why Don't You Have QoS 0?

MQTT v5's flow control mechanism relies on response packets, limiting flow control to QoS 1 and 2 messages. This is because QoS 0 messages do not generate response packets(Frustaci, Pace et al. 2017). However, an alternative proposal suggests that when the sending quota reaches zero, the sender may either continue sending PUBLISH packets with QoS 0 or suspend sending altogether. Suspending sending can be beneficial when QoS 1 and 2 PUBLISH packets face delays in response times, indicating a potential reduction in the receiver's processing capability (Frustaci, Pace et al. 2017, Qiu, Tian et al. 2020).

5.5 Summary

Despite some limitations, it is recommended that users employ MQTT v5's flow control mechanism. The sending quota algorithm, based on response packets, optimally utilizes sender resources. The Receive Maximum attribute enhances transparency and flexibility, particularly in scenarios involving multi-vendor equipment (Bashir and Mir 2020).

6. CONCLUSION

This paper provides an overview of the MQTT protocol, its strengths and weaknesses with respect to security. In conclusion, despite some shortcomings in the flow control mechanism of MQTT, it is highly recommended its utilization for users seeking efficient and flexible IoT communication. MQTT's innovative sending quota algorithm, reliant on response packets, optimizes resource utilization for senders. The introduction of the Receive Maximum attribute eliminates the need for preemptive sending quota negotiations, fostering transparency and adaptability, particularly in diverse equipment scenarios.

To recap the key points discussed in this article, it's crucial to recognize the drawbacks associated with the MQTT protocol, which include:

1. **Protocol Choice:** MQTT relies on the TCP protocol, which can increase processing power and memory usage. The frequent wake-up and communication intervals in the TCP handshake can lead to higher battery consumption.
2. **Scalability:** The use of a centralized broker can sometimes limit scalability due to the overhead it imposes on each client device. Utilizing a local broker hub can be an effective way to overcome this challenge.
3. **Single Point of Failure:** A centralized broker can introduce a single point of failure, as client connections with the broker remain open. It's essential to consider redundancy and fault tolerance in your MQTT deployment.
4. **Implementation Complexity:** Compared to HTTP, MQTT implementation can be more challenging, especially for those new to the protocol.
5. **Advanced Features:** MQTT may lack support for advanced features like built-in flow control, which is an essential consideration for specific use cases.
6. **TCP/IP Requirement:** Clients using MQTT must support TCP/IP, which may limit compatibility in some scenarios.

In light of these considerations, MQTT remains a powerful protocol for IoT communication, but careful planning and evaluation of its features and drawbacks are necessary to make the most of this technology. By addressing these challenges effectively, users can harness MQTT's strengths to build robust and efficient IoT systems.

7. REFERENCES

Al-Hawawreh, M. and E. Sitnikova (2020). Developing a security testbed for industrial internet of things. *IEEE Internet of Things Journal* 8(7): 5558-5573.

Andy, S., Rahardjo, B., &Hanindhito, B. (2017, September). Attack scenarios and security analysis of MQTT communication protocol in IoT system. In *2017 4th international conference on electrical engineering, computer science and informatics (EECSI)* (pp. 1-6). IEEE.

Atmoko, R. A., Riantini, R., & Hasin, M. K. (2017, May). IoT real time data acquisition using MQTT protocol. In *Journal of Physics: Conference Series* (Vol. 853, No. 1, p. 012003). IOP Publishing.

Bashir, A., & Mir, A. H. (2020). Lightweight Secure-MQTT for Internet of Things. In *Optical and Wireless Technologies: Proceedings of OWT 2019* (pp. 57-66). Springer Singapore.

Bender, M., Kirdan, E., Pahl, M. O., & Carle, G. (2021, January). Open-source MQTT evaluation. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)* (pp. 1-4). IEEE.

Bhawiyuga, A., Data, M., & Warda, A. (2017, October). Architectural design of token based authentication of MQTT protocol in constrained IoT device. In *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)* (pp. 1-4). IEEE.

Dinculeană, D., & Cheng, X. (2019). Vulnerabilities and limitations of MQTT protocol used between IoT devices. *Applied Sciences*, *9*(5), 848.

Fauzan, A., Sukarno, P., & Wardana, A. A. (2020, September). Overhead Analysis of the Use of Digital Signature in MQTT Protocol for Constrained Device in the Internet of Things System. In *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)* (pp. 415-420). IEEE.

Ferrari, P., Flammini, A., Sisinni, E., Rinaldi, S., Brandão, D., & Rocha, M. S. (2018). Delay estimation of industrial IoT applications based on messaging protocols. *IEEE Transactions on Instrumentation and Measurement*, *67*(9), 2188-2199.

Froiz-Míguez, I., Fernández-Caramés, T. M., Fraga-Lamas, P., & Castedo, L. (2018). Design, implementation and practical evaluation of an IoT home automation system for fog computing applications based on MQTT and ZigBee-WiFi sensor nodes. *Sensors*, *18*(8), 2660.

Frustaci, M., Pace, P., Aloi, G., & Fortino, G. (2017). Evaluating critical security issues of the IoT world: Present and future challenges. *IEEE Internet of things journal*, *5*(4), 2483-2495.

Grgić, K., Špeh, I., &Heđi, I. (2016, October). A web-based IoT solution for monitoring data using MQTT protocol. In *2016 international conference on smart systems and technologies (SST)* (pp. 249-253). IEEE.

Gupta, V., Khera, S., & Turk, N. (2021). MQTT protocol employing IOT based home safety system with ABE encryption. *Multimedia Tools and Applications*, *80*(2), 2931-2949.

Hiromoto, R. E., Haney, M., &Vakanski, A. (2017, September). A secure architecture for IoT with supply chain risk management. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* (Vol. 1, pp. 431-435). IEEE.

Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008, January). 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)* (pp. 791-798). IEEE.

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., & Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud computing*, *3*(1), 11-17.

Kumar, A., Sharma, S., Goyal, N., Singh, A., Cheng, X., & Singh, P. (2021). Secure and energy-efficient smart building architecture with emerging technology IoT. *Computer Communications*, *176*, 207-217.

Kumari, A., Tanwar, S., Tyagi, S., & Kumar, N. (2018). Fog computing for Healthcare 4.0 environment: Opportunities and challenges. *Computers & Electrical Engineering*, *72*, 1-13.

Lara, E., Aguilar, L., Sanchez, M. A., & García, J. A. (2020). Lightweight authentication protocol for M2M communications of resource-constrained devices in industrial Internet of Things. *Sensors*, *20*(2), 501.

Liu, X., Zhang, T., Hu, N., Zhang, P., & Zhang, Y. (2020). The method of Internet of Things access and network communication based on MQTT. *Computer Communications*, *153*, 169-176.

Obaidat, M. A., Obeidat, S., Holst, J., Al Hayajneh, A., & Brown, J. (2020). A comprehensive and systematic survey on the internet of things: Security and privacy challenges, security frameworks, enabling technologies, threats, vulnerabilities and countermeasures. *Computers*, *9*(2), 44.

Qiu, J., Tian, Z., Du, C., Zuo, Q., Su, S., & Fang, B. (2020). A survey on access control in the age of internet of things. *IEEE Internet of Things Journal*, *7*(6), 4682-4696.

Radwan, N. M. (2020). A study: The future of the internet of things and its home applications. *International Journal of Computer Science and Information Security (IJCSIS)*, *18*(1).

Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., & Liljeberg, P. (2018). Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*, *78*, 641-658.

Ray, P. P. (2017). Internet of things for smart agriculture: Technologies, practices and future direction. *Journal of Ambient Intelligence and Smart Environments*, *9*(4), 395-420.

Soni, D., & Makwana, A. (2017, April). A survey on MQTT: a protocol of internet of things (IOT). In *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)* (Vol. 20, pp. 173-177).

Swamy, S. N., Jadhav, D., & Kulkarni, N. (2017, February). Security threats in the application layer in IOT applications. In *2017 International conference on i-SMAC (iot in social, mobile, analytics and cloud)(i-SMAC)* (pp. 477-480). IEEE.

Upadhyay, Y., Borole, A., &Dileepan, D. (2016, March). MQTT based secured home automation system. In *2016 Symposium on Colossal Data Analysis and Networking (CDAN)* (pp. 1-4). IEEE.

Vongsingthong, S., &Smanchat, S. (2014). Internet of things: a review of applications and technologies. *Suranaree Journal of Science and Technology*, *21*(4), 359-374.

Wang, Q., Ji, S., Tian, Y., Zhang, X., Zhao, B., Kan, Y., Lin, Z., Lin, C., Deng, S., Liu, A.X. & Beyah, R. (2021). MPInspector: A systematic and automatic approach for evaluating the security of IoT messaging protocols. In *30th USENIX Security Symposium (USENIX Security 21)* (pp. 4205-4222).

Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S., & Al-Hatmi, R. (2017, May). Internet of Things: Survey and open issues of MQTT protocol. In *2017 international conference on engineering & MIS (ICEMIS)* (pp. 1-6). IEEE.

8. Appendix 1

TABLE 1: Public MQTT Brokers.

Name	Broker Address	TCP Port	TLS Port	Web-Socket Port	Message Retention	Persistent Session	Sign Up Requred
Eclipse	mqtt.eclipse.org	1883	N/A	80, 443	Yes	Yes	No
Mosquitto	test.mosquitto.org	1883	8883, 8884	80	Yes	Yes	No
HiveMQ	broker.hivemq.com	1883	N/A	8000	Yes	Yes	No
Flespi	mqtt.flespi.io	1883	8883	80, 443	Yes	Yes	Yes
D(Internet of Things)y	mqqt.d(Internet of Things)y.co	1883	8883	8080, 8880	Yes	Yes	Yes
Fluux	mqtt.fluux.io	1883	8883	N/A	N/A	N/A	No
Emqx	Broker.emqx.io	1883	8883	8083	Yes	Yes	No

TABLE 2: Private Brokers.

Name	Link	TCP Port	TLS Port	Web-Socket Port	Message Retention	Persistent Session	QoS Levels	Free Limits	Link
Azure	Link	No	8883	443	No	Limited	0, 1	8000 messages/day	Link
AWS	Link	No	8883	443	No	Limited	0, 1	250,000/month	Link
CloudMQTT	Link	Custom Port	Custom Port	Custom Port	Not Sure	Yes	0, 1, 2	5Connections & 10 Kbit/s	Link