Master's thesis

Master’s Programme in Embedded and Intelligent Systems, 120 credits
Masters in Information Technology,120 credits

LP_MQTT

A Low-Power IoT Messaging Protocol Based on MQTT Standard

Computer Science and Engineering, 30 credits

Halmstad, 02-02-2024
Anchu Antony, Deepthi Myladi Kelambat

HALMSTAD UNIVERSITY

ABSTRACT

In the Internet of Things (IoT) era, the MQTT Protocol played a big part in increasing the flow of uninterrupted communication between connected devices. With its functioning being on the publish/subscribe messaging system and having a central broker framework, MQTT considering its lightweight functionality, played a very vital role in IoT connectivity. Nonetheless, there are challenges ahead, especially in energy consumption, because the majority of IoT devices operate under constrained power sources. In line with this, our research suggests how the MQTT broker can make an intelligent decision using an intelligent algorithm. The algorithm idealizes wake-up times for subscriber clients with the aid of previous data, including machine learning (ML) regression techniques in the background that produce substantial energy savings. The study combines the regression machine learning approaches with the quality of service levels' incorporation into the decision framework through the introduction of operational modes designed for effective client management. The research, therefore, aims universally to enhance the efficiency available in MQTT making it applicable across diverse IoT applications by simultaneously addressing both the broker and the client sides . The versatile approach ensures more performance and sustainability for MQTT, further strengthening its build as one of the building blocks for energy efficient and responsive communication in the IoT. Deep learning approaches that follow regression will be the required leap for the transformation of energy consumption and adoption of resource allocation within IoT networks to an optimization level that would unlock new frontiers of efficiency for a sustainable connected future.

ACKNOWLEDGEMENTS

First and foremost, We extend our deepest gratitude to God, for bestowing upon us the strength, wisdom, and serenity throughout this challenging yet rewarding journey. We are immensely grateful to our family, whose unwavering support, endless love, and constant encouragement have been our pillars of strength. Their sacrifices have not gone unnoticed, and We owe a great deal of our success to their unyielding faith in us. A special note of thanks to our supervisor, Mahdi Fazeli, whose expertise and insightful guidance have been instrumental in shaping our research. his patience, knowledge, and commitment to excellence have profoundly influenced our work and personal growth. We would also like to express our sincere appreciation to the examiner Slawomir Nowaczyk and the opponent Hazem Ali, whose rigorous reviews and constructive Feedback have significantly contributed to the refinement and depth of this thesis. This journey has been a testament to the power of collaboration, guidance, and familial support. To everyone who played a part, no matter how small, We are eternally grateful.

Thank you,

Anchu Antony & Deepthi Myladi Kelambath

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

LP-MQTT Low Power Messege Query Telementry Transport

MQTT Messege Query Telementry Transport

MLP Multilayer Preceptron

KPI Key Performance Indicator

Part I

Here presents an overview, introducing the primary challenge under investigation. Briefly defining the problem, framing it within a broader context, and emphasizing its significance. This sets the foundation for a detailed examination and discussion of potential solutions in the following sections.

1 INTRODUCTION

The development of the Internet of Things (IoT) introduced in an age in which countless levels of connectivity between physical objects share volumes of data. The MQTT Protocol, an excellent standard for communication that has revolutionized the way IoT devices interact and communicate, takes place at possibly the lowest level. Initially developed for low-bandwidth and satellite communications, the MQTT Protocol uses a publish/subscribe messaging mechanism when using a central server known as a broker that makes it possible for different clients to communicate. When desired client messages are labeled with respective topics, the broker informs other clients subscribed to which have corresponding topics and efficiently delivers the message to them.

MQTT is being a many-to-many protocol, helps deliver flawless transmission of messages between clients from the broker, forming a base of IoT connectivity. The immense popularity of the MQTT Protocol within IoT applications lies in its inherent advantages. Its low complexity structure makes it suitable for a wide range of devices. Furthermore, MQTT's power-efficient design aligns well with the constraints of IoT devices. This supports bidirectional communication and enables effective two-way data interchange. Its scalability to support a vast number of devices, together with reliable one-time message delivery despite unreliable networks, MQTT thus positions the IoT as a feasible opportunity that needs to be realized. In addition, minimal resource consumption, possibly due to the protocol's small footprint, is another critical feature for IoT devices with constrained computational capabilities.

Actual applications of the MQTT Protocol span numerous domains. From remote sensing and smart cities to social media platforms and home automation, the protocol underpins the operation of intelligent farming, wearable technology, manufacturing processes, and even oil and gas industries. Amid these vivid fields of application, power consumption poses a formidable challenge. The dependence on battery power or the limited availability of an energy source accentuates the pressing need to minimize power consumption. Extending the lifespan of batteries and reddening the necessity for frequent replacements or recharges is imperative to the sustainability of IoT deployments.

The geographical diversity of IoT applications also introduces scenarios where the devices must operate in remote, inaccessible locations with minimal or no power availability. This accentuates the sig-

nificance of efficient power management strategies to ensure uninterrupted device operation and data exchange. This research, therefore, proposes a new approach to tackle these challenges and enhance the energy efficiency of MQTT-enabled IoT networks. The basis of this approach would be to increase decision-making capabilities within the MQTT broker. An intelligent algorithm is envisaged to achieve such a goal, harnessing historical data of topic activity across the network. This algorithm aspires to determine, utilizing data-driven insights, optimal wake-up times for subscriber clients. Allowing these clients to enter low-power modes during reduced topic activity saves substantial energy. Achieve fostering a world of accelerated efficiency. In an MQTT-enabled IoT network, two main entities shape the landscape: the message broker and the clients. Clients take up the roles of publishers as well as subscribers where. In contrast, the broker presides over between the clients the delivery of the packets 'delivery between clients, Ending the broker with an effective decision-making algorithm that is advanced and optimized to utilize transceiver modules for publisher clients. This results in a paradigm shift towards more efficient energy consumption within the network.

Central to the algorithm's operation are the QoS levels supplied dynamically by the MQTT protocol. Changing QoS levels in its classification to QoS 0, QoS 1, and QoS 2 are critical in ensuring the delivery of messages. QoS 0 is efficient at cost potential message loss, QoS 1 guarantees that at least one copy of each message will be delivered, and QoS 2 assumes accurate delivery of messages. QoS level choice is adjusted to the needs specific to the application being performed and possible undertakings of respective devices.

A two-time investigation by the study is needed to enhance energy efficiency. First, We explored the possibility of integrating machine learning techniques within the broker's decision-making alloy. With the help of Deep Learning, the algorithm can learn itself and adjust to network conditions. Refine clients' wake-up times and energy saving in real-time. The second avenue delves into incorporating QoS levels into the decision-making algorithm. Here, incorporating QoS levels enhances the algorithmâs capacity to choose high-priority message delivery while having energy-efficient wake-up times for subscriber clients.

The two operational modes, "idle mode" and "running mode," efficiently control clients' behavior. Idle mode represents a "sleep mode" whereby there is low-power standby of clients awaiting specific events. Conversely, the running mode is data transmission and reception, with QoS levels governing reliability and message delivery guarantees. Contemporary power-saving techniques mainly target the device side. However, this research aims for a comprehensive approach addressing both the broker and client sides. The exploration encompasses sleep modes, keep-alive interval adjustment, reduction in mes-

sage publication, payload compression, and dynamic alteration of QoS levels. The decision-making algorithm is probably implemented to augment the broker's protocol efficiency further.

Power-saving techniques may, however, bring challenges such as increased latency or necessary implementation reliability that could not be compatible with the requirements of applications. Moreover, hardware or software limitations can restrict the application of these techniques, mainly for devices that require continuous connectivity. Finally, this research aims to make the MQTT protocol applicable to any device worldwide, regardless of future size. On the other hand, the dual strategies of incorporating QoS levels and introducing the âSleep modeâ level are expected to yield reductions in latency and an enhancement of reliability. Owing to the number of MQTT clients or publishers, these contribute to the network by generating data aligned with their specific applications or use cases.

Data produced by such clients can assume periodic patterns, for example, when monitoring sensor data at time intervals or stochastic patterns for processing data from mobile devices showing different usage behaviors. Most importantly, is the fact that the data produced by such clients or publishers also manifests as a fusion, the fusion of both periodic and stochastic patterns. For instance, a device monitoring a production line might generate regular sensor readings as periodic data and spontaneous alarms triggered by unexpected events as stochastic data. This emphasizes the fundamental event nature of driven MQTT data production, where the data occurs upon the occurrence of certain events rather than being constrained to fixed temporal intervals. These events can be influenced by an array of factors that may or may not include shifts in the environment, user interactions, and the device's operational status.

This research aims to enhance an energy-efficient environment within MQTT-enabled IoT networks and embarks on a multi-faceted journey. The interweaving of machine learning techniques, integration of QoS levels, and implementation of new power-saving strategies are attempted to improve the performance and sustainability of this protocol across diverse IoT applications. This initiative is poised to strengthen MQTT's positioning as a quintessential IoT communication protocol, facilitating interconnected landscapes centric to energy yet highly responsive.

In contemporary data science, deep learning has emerged as a disruptive paradigm that empowers many machines to extract intricate patterns and representations from vast datasets. A subset of machine learning, deep learning operates through neural networks with multiple layers, enabling the discovery of intricate relationships that conventional algorithms might overlook. This revolutionary approach has witnessed remarkable success across domains like computer natural language processing, vision, and speech recognition.

Deep learning can potentially revolutionize decision-making algorithms in the context of the MQTT protocol's energy efficiency. By applying a deeper neural network, the algorithm would learn from topic activity historical data while unearthing some nuances and adapting and evolving network dynamics. The deep learning capability to recognize the complex patterns inside the data may allow precise prediction of client wakeup time for optimization energy consumption patterns and, as a consequence, will improve protocol efficiency.

Moreover, machine learning has potential employment in developing advanced QoS-level prediction models. From learning from historical data and considering the contextual intricacies, the number of trends, dependencies, and anomalies influence the choice of a QoS level. machine learning models can, in the process of learning, from historical data and considering the contextual intricacies of the network, identify several trends, dependencies, and anomalies that shape the choice of QoS levels.

This approach would hence have energy-efficient wake-up times harmonized with delivering high-priority messages, enabling a compromise between reliability and efficiency. Integrating deep learning into the decision-making algorithm of the MQTT protocol will require creating robust training datasets, designing appropriate neural architectures, and fine-tuning model parameters. The research aims to leverage the power of -taking MQTT-enabled IoT networks to an era of improved energy efficiency, reliability, and responsiveness.

In conclusion, combining deep learning techniques with MQTT's decision-making algorithm presents a radical move towards enhance energy consumption and resource allocation within IoT networks. This combination is cutting-edge in that it produces edge technology and communication protocols that promise to unlock new frontiers of efficiency, enabling the IoT ecosystems to thrive in a sustainable and connected future.

2

PROBLEM STATEMENT

We propose to improve the decision-making capabilities of the MQTT broker by implementing an intelligent algorithm that considers the published history of topics on the network. The algorithm will determine the optimal wake-up time for subscriber clients, allowing them to conserve energy by entering low-power modes during periods of low-topic activity. This approach will improve the overall energy efficiency of the MQTT-enabled IoT network. In an MQTT-enabled IoT network, there are two types of entities: the message broker and the clients. Clients can act as publishers or subscribers of topics while the broker manages packet deliveries between clients by equipping the broker with this improved decision-making algorithm. Decision-making algorithms make decisions based on dynamically changing Quality of Service (QoS) levels, which are determined by analyzing published history.

This study addresses the inherent limitations of the traditional MQTT protocol, specifically its static Quality of Service (QoS) levels, which can lead to suboptimal energy consumption and make it less suitable for high-end devices. The fixed QoS levels restrict adaptability to varying environmental conditions, resulting in inefficiencies.

To improve the energy efficiency of MQTT, we propose a novel approach that dynamically adjusts QoS levels based on predictive analytics. This dynamic adaptation aims to optimize the trade-off between reliability and performance in communication between clients and brokers. Our approach allows real-time decision-making, enabling the protocol to seamlessly adapt to changing conditions and evaluate communication success in diverse environmental scenarios.

We utilize historical data to predict QoS levels by employing machine learning techniques, explicitly considering clients' wake-up time and idle time. This predictive model enhances the energy efficiency of MQTT by tailoring QoS levels to the specific requirements of different environmental scenarios. Through this innovative approach, we demonstrate the potential to significantly improve the adaptability and performance of MQTT in high-end devices.

Our thesis will try to solve this problem by developing an intelligent decision-making algorithm. This algorithm will be integrated into the MQTT broker and utilize machine learning techniques to analyze previous activity data related to specific topics. The aim is to improve the power efficiency of MQTT in IoT devices by dynamically assigning QoS levels based on the failure rate of the environments; we incorporate this to put clients in standby mode as much as pos-

sible. We have proposed a dynamic QoS-level assignment based on environmental conditions. Also, to predict the ideal time of the client during running mode to keep the client in sleep mode to minimize the power consumption when the client is not doing proper computation. We explored two dynamic approaches: the Statical Dynamic and MLPRegressor approaches.

Part II

Part 2 involves a comprehensive literature analysis to examine and evaluate previous research on the management of Quality of Service (QoS). Subsequently, a QoS management policy is presented, specifically designed to target and resolve identified deficiencies. Following that, the outcomes of adopting this policy are presented and analyzed, resulting in findings that highlight the beneficial effects of the approach and possible areas for enhancement.

3 LITERATURE REVIEW

3.1 COMPARISON ON DIFFERENT PROTOCOLS

We reviewed papers related to the comparative analysis of MQTT with other protocols. We found specific scope, benefits, as well as drawbacks while using the MQTT protocol over other protocols we have gone through [1],[18],[5],[8],[14],[16],[21],[22],[25]

The authors of the above papers say that MQTT is designed for environments with limited bandwidth and computing resources, which makes MQTT suitable for constrained IoT devices.MQTT is good at handling high-volume messaging and high-traffic networks, which helps MQTT remain a lightweight protocol and efficient in message delivery.MQTT supports various quality of service to ensure reliable message delivery.

Compared with COAP, MQTT has some drawbacks, such as high latency and bandwidth consumption; thus, COAP overcomes MQTT. But MQTT shows better performance in high reliability and large message size and also less delay for higher traffic networks, So MQTT overcomes COAP in performance in lower delay but with higher power Consumption MQTT compared with AMQP both protocols work on Publish/Subscribe model, but MQTT is better than AMQP in message delivery and Payload limit.

When comparing MQTT and HTTP, it says that HTTP has higher power consumption than MQTT due to overhead, so HTTP is unsuitable for small and frequent messages, whereas IOT always needs frequent messages. HTTP works on the pull model, which is the client's need to initiate a request for new messages. In contrast, MQTT works on the Publisher/Subscriber model, so HTTP is unsuitable for real-time scenarios, and IoT requires continuous monitoring and immediate updates, such as sensor networks. HTTP is less power efficient than MQTT, so HTTP cannot be used in remote areas or with less limited power sources.

HTTP has no built-in QoS levels, so there is no confirmation in HTTP that messages have been delivered. In contrast, MQTT has QoS levels that ensure message delivery on the requirements of applications, even in network problems.HTTP can't be used for large-scale environments because it is suitable only in a traditional web application where frequent messaging is not required. Still, the MQTT protocol can be used on a large scale in an IoT environment due to its lightweight and Publisher/Subscriber model.

A study compared the suitability of the MQTT, CoAP AMPQ, and HTTP messaging protocols in IoT systems. Each protocol was analyzed for strengths, limitations, and performance characteristics such as message size, overhead, resource and power requirements, latency, and bandwidth.CoAP has the lowest power consumption, HTTP has the highest message size and overhead, MQTT has the best QoS support, and AMQP is preferred for business applications. Although CoAP has the lowest power consumption in static environments in dynamic network conditions, CoAP is suitable and can also be used in low-scale environments. The reliability of CoAP is lower in large-scale environments.

Studying different papers on comparative studies of different protocols makes us understand MQTT more deeply than other protocols. It also helps to understand why we need to enhance the power efficiency of MQTT for IoT devices.

3.2 STUDIES ON DIFFERENT POWER SAVING TECHNIQUES OF MQTT

Researchers proposed sleep mode, Keep-alive arrival adjustments, message compression, and QoS dynamic change but have not done adequate studies while going through the paper [22]. Sleep Mode in IoT devices represents a crucial strategy for energy conservation, particularly relevant in the context of wireless sensor networks (WSNs). A device significantly reduces power consumption in this operational mode by deactivating its communication modules and other non-essential functions. This approach is essential for prolonging the operational lifespan of battery-powered devices, which constitute a significant portion of IoT deployments. Regular cycles of activity and inactivity, governed by sleep mode algorithms, ensure that devices perform necessary tasks, such as data transmission, at optimal intervals, thereby balancing energy efficiency with functional efficiency.

The optimization of Keep Alive intervals [14] in network protocols like MQTT is a subject of ongoing research, focusing on the balance between connection responsiveness and network resource utilization. These intervals determine the frequency at which 'heartbeat' signals are sent to confirm the integrity of network connections. Shorter intervals can lead to increased network traffic and power usage, while longer intervals may delay the detection of connection losses, impacting the reliability of IoT systems. Optimal Keep-Alive interval settings are context-dependent, balancing the need for timely communication with power and bandwidth availability constraints.

Message compression [3] is increasingly recognized as a vital technique in IoT communication, particularly for bandwidth-constrained environments. Compression techniques facilitate faster transmission speeds and lower bandwidth consumption by reducing the size of data packets. This is especially beneficial when IoT devices rely on

limited or expensive communication mediums like cellular networks. The researcher highlights that effective message compression conserves network resources and reduces energy consumption in data transmission, a key consideration for battery-dependent IoT devices.

The concept of Dynamic Quality of Service (QoS)[12] adjustment in IoT networks addresses the need for flexible communication strategies in varying network conditions. In protocols like MQTT, different QoS levels offer varying message delivery guarantees, from 'at most once' to 'exactly once.' The ability to dynamically adjust these QoS levels allows IoT systems to adapt to the current state of the network, the device's energy levels, and the message content's importance. This flexibility is essential for efficient network resource management, ensuring the system allocates more resources for critical communications while conserving them for less critical transmissions.

These studies help us know about QoS levels, keep alive intervals, message compression, and sleep mode. In this paper[22] discussed only about Sleep mode. This paper[14] discusses only about keep alive intervals, and this paper[12] conducts studies on different QoS levels. All these studies' authors use any of the power-saving techniques. In our thesis, we are using sleep mode, keep alive intervals, and QoS levels. We are using these techniques in wide networks as well as different environmental conditions, thus enhancing the power efficiency of MQTT.

3.3 STUDIES FOCUS DEVICE CENTRIC

Studies like [10], [13] focus on clients in MQTT protocol. Studies on optimizing in Broker are rare to get. The broker is the central server that manages the delivery of messages. Optimizing the broker is also promising research to focus on. Techniques for message compression, QoS management, adaptive data transmission strategies, and network-aware MQTT client behavior. Our thesis aims to focus on both the broker and client sides to enhance MQTT's power efficiency.

3.4 STUDIES FOCUS ENERGY CONSUMPTION OF MQTT

There are papers focusing on and comparing the energy consumption of MQTT: [19],[3]. MQTT and CoAP power consumption were studied, CoAP being suited to energy-constrained devices, but MQTT being high power consumption due to the use of TCP. The power consumption of CoAP is low when it is used in low bandwidth and sending small messages, as well as when it depends on the network environment. The advantage of MQTT over CoAP is that it is more efficient and reliable in sending large-size messages. We propose to enhance the power efficiency of MQTT with the help of QoS levels

that can be used in various network environments. Previous studies have shown how much energy different protocols consume within one network environment. MQTT can be made low power with the help of intelligent algorithms and QoS levels in various network environments.

3.5 STUDIES ON MACHINE LEARNING/DEEP LEARNING EXPLORATION

We have tried to find papers on optimizing the MQTT using machine learning and deep learning. However, we can find machine learning or deep learning on the Security of MQTT, which is far from our proposed approach. Papers discussed in [12] general features of MQTT like delay parameters of MQTT, general information on IoT ideas, which make us gather more information on MQTT, the security challenge of MQTT[6].

Our studies focus on how to enhance the power efficiency of MQTT using machine learning algorithms. How previous data can help decide the wake-up and sleep mode of MQTT, thus reducing power consumption in MQTT.

We have done a literature review on 25 papers related to the MQTT Protocol. Even though we found only a few papers related to Energy consumption of MQTT Protocol. We have compared power efficiency, reliability, and response time. A table has been created on features below Table 1

Table 1: Comparing Methods in Literature Survey

Paper	Method/Features	Power Efficiency	Response Time	Reliability
[1]	MQTT Protocol	Low power consumption	High latency.	High reliability
	COAP Protocol	Efficient power usage	Low latency	Moderate reliability
	AMQP Protocol	Energy-inefficient	Variable depends on	Robust performance
	Enhancements to MQTT (Temporal Data and Aggregation)	Not specified	Reduced communication	Not specified
[2]	Lightweight Generating and Key Exchange	High	Low	High
[3]	MQTT Protocol	High	Low	High
	Intelligent active	Improved	Reduced	High
	CSMA/CA with Time Slot Division	Improved	Reduced	High

Continued on next page

Table 1 – continued from previous page

Paper	Method/Features	Power Efficiency	Response Time	Reliability
	Route Discovery	Not specified	Not specified	Not specified
	Data Sharing	Not specified	Not specified	Not specified
[4]	MQTT QoS "At most once" over TLS	High	Fast	Medium
	MQTT QoS "At least once" over TLS	Medium-High	Fast	High
	MQTT QoS "Exactly once" over TLS	Medium	Slow	High
[6]	Machine Learning-based MQTT Security	High (Random Forest performs better)	Not specified	Not specified
	Replay Attack Prevention (ECC, Timestamp, Wake-up Mechanism)	Not specified	Not specified	Not specified
	MITM Attack Resistance	Not specified	Not specified	Not specified
	Anomaly Detection and Vulnerability Analysis	Not specified	Not specified	Not specified
	MQTT Network Mutual Trust Mechanism supported by Blockchain	Not specified	Not specified	High
	Encrypted Transmission of MQTT Network	Not specified	Not specified	High
	DoS Attack Prevention (SecureMQTT, Throttling)	Varies	Varies	Varies
[7]	Value-to-HMAC Mapping	Efficient	Fast	High
[11]	Low-voltage terminals plugged into SDT	High	Within 1 minute	High
	SDT plugged into the master station	High	Not specified	High

Continued on next page

Table 1 – continued from previous page

Paper	Method/Features	Power Efficiency	Response Time	Reliability
	Applications plugged into the master station	High	Not specified	High
	Mesh network module (for low-voltage term.)	High	Not specified	Not specified
	PLC communication module (for low-voltage term.)	Not specified	Not specified	Not specified
	Embedding internal communication module (for primary equipment)	High	Not specified	High
[15]	MQTT	High	Not specified	High
	CoAP	High	Not specified	High
	RESTful HTTP	Moderate	Not specified	Moderate
[16]	MQTT	High	Low	High
	CoAP	High	Low	Moderate
	AMQP	Moderate	Moderate	High
	HTTP	Low	High	High
[17]	Data Transfer (CoAP vs. MQTT)	Comparable for small packets; MQTT favored for larger packets	Constant for small packets; Higher penalty for CoAP with larger packets	Depending on the use case, MQTT is more efficient for larger payloads
	DTLS Handshake (RSA, ECDSA, ECDHE, DHE)	Varies based on cipher suite and key exchange method	Slow, in the order of 50-500 Wh	DHE is less efficient than ECDHE; RSA performs better without client authentication
	Firmware Update (AES Key Sizes, Modes)	Aggregating data is beneficial; Slightly less overhead for CCM8	Not specified	Not specified

Continued on next page

Table 1 – continued from previous page

Paper	Method/Features	Power Efficiency	Response Time	Reliability
	Packet Loss (CoAP vs. MQTT)	Lossy networks Favor MQTT (up to 91% energy savings at a 20% loss rate)	Not specified	Not specified
[19]	MQTT-SN with ChaCha20-Poly1305 AEAD	High	Low	High
[20]	Publish/Subscribe	High (low overhead)	Low	High
	QoS Levels	Varies	Varies	Varies
	Lightweight Protocol	High	Low	High
	Connectivity (Suitable for M2M, IoT)	Not specified	Depends on implementation	High
	MQTT Broker	Varies by implementation	Depends on the broker	Varies by broker
	Real-world Apps	Varies by application	Varies by application	Varies by application
[22]	UDP	High	Low	Low
	TCP	Moderate	Moderate	High
	CoAP (over UDP)	High	Moderate	Moderate
	CoAP (over TCP)	Moderate	Moderate	High
	MQTT (over TCP)	Moderate	Moderate	High
	MQTT (over UDP)	High	Moderate	Moderate
[23]	Lightweight API	Requires minimal power	Quick	Reliable
	Publish-Subscribe Model	Low battery consumption	Low latency	Many-to-many communication
[24]	Lightweight MQTT Protocol	Low power consumption	Low latency	Reliable data transfer
	Quality of Service Levels (QoS)	Depends on QoS level	Varies with QoS	Acknowledgment for reliability
	MQTT-SN Protocol for Sensor Networks	Energy-efficient	Minimal delay	Ensured message delivery

Continued on next page

Table 1 – continued from previous page

Paper	Method/Features	Power Efficiency	Response Time	Reliability
	Battery-Powered IoT Devices	Energy consumption varies with QoS levels	Low latency	Assured message delivery
	MQTT Protocol	Low	Longer latency	Limited reliability
	HTTP Protocol	High	Not specified	Not specified
[25]	MQTT	Low	Fast	High (Guaranteed delivery options)
	HTTP	High	Slow	Lower (No assured delivery by default)
[18]	Continuous Data Transmit (MQTT)	High	Low (Real-time)	High
	Continuous Data Transmit (HTTP)	Moderate	Low (Real-time)	High
	Periodic Data Transmit (MQTT)	High	Moderate (Interval)	High
	Periodic Data Transmit (HTTP)	Moderate	Moderate (Interval)	High
	Periodic Data Transmit with Deep Sleep (MQTT)	High	Low (Interval + RTC)	High with power savings
	Adaptive Data Transmit (MQTT)	High	Variable (Based on change)	High
	Adaptive Data Transmit (HTTP)	High	Variable (Based on change)	High
[14]	Protocol Integration	Features designed to improve power efficiency, including low-power modes and optimized data transmission	Features minimizing response times, prioritizing critical data, and reducing latency in message delivery	Features ensuring enhanced reliability, such as acknowledgment mechanisms and error recovery strategies
	Message Queuing Optimization			

Continued on next page

Table 1 – continued from previous page

Paper	Method/Features	Power Efficiency	Response Time	Reliability
	Data Compression Techniques Efficiency Load Balancing Strategies			
[5]	Integration of Emerging IoT building functionality Technologies	Strategies for managing the power consumption of sensors to extend their lifespan and optimize energy usage.	Not Specified	Not Specified
	Secure Communication Protocols within the smart building architecture	Implementation of adaptive systems for lighting and HVAC to optimize energy consumption based on occupancy and environmental conditions.		
	Energy-Efficient Infrastructure actuators, and communication devices	Integration of energy harvesting technologies to capture and utilize ambient energy sources for powering IoT devices. |		
[9]	MQTT	Varies based on QoS	Low latency for message delivery	High, QoS-dependent
	CoAP	Efficient for resource-constrained	Low latency for message delivery	Moderate to high

Continued on next page

Table 1 – continued from previous page

Paper	Method/Features	Power Efficiency	Response Time	Reliability
	OPC UA	Resource-intensive	Variable depending on implementation	High
[21]	QoS0	High	Low	Low
	Qos1	Moderate	Moderate	Moderate
	Qos2	Low	High	High
[10]	QoS Levels (0, 1, 2)	Not specified	Not specified	High
	MQTT Protocol	Not specified	Not specified	High
	Wired Network Analysis	Moderate to High	Varies	High
	Wireless Network Analysis	Moderate to High	Varies	Moderate
	Correlation Analysis	Not specified	Not specified	Strong
[13]	IoT Electronic Platform (HTTP)	Lower power efficiency (667.33 mW)	Fast transmission (not specified)	Not specified
	IoT Electronic Platform (MQTT, QoS 0)	Moderate power efficiency (629.68 mW)	Fast transmission (not specified)	Fast transmission (not specified)
	IoT Electronic Platform (MQTT, QoS 1)	High power efficiency (614.33 mW)	Fast transmission (not specified)	High reliability with QoS 1
	IoT Electronic Platform (MQTT, QoS 2 - Future Work)	Not specified	Not specified	Not specified
[8]	MQTT vs. HTTP Power Consumption Comparison	MQTT (QoS 1) more power-efficient	Not specified	Not specified
	MQTT Device-to-Gateway Model	No energy wasted in cloud-based broker	Not specified	Not specified
	Prediction Model for Battery Life Cycle	Not specified	Increase in error percentage over time	Not specified

Continued on next page

Table 1 – continued from previous page

Paper	Method/Features	Power Efficiency	Response Time	Reliability
	Strategies to Optimise Power Consumption in IoT Systems	Varies based on strategies	Not specified	Not specified
[12]	MQTT Protocol	Not mentioned	Varies with payload size	Varies with QoS levels
	QoS Levels (0, 1, 2)	Not mentioned	Level-dependent	Level-dependent
	Delivery Modes (Plaintext, TLS v1.3, TLSMA)	Varies with encryption	Varies with encryption	Varies with encryption
	Payload Size	Not mentioned	Increases communication delay	Affects communication delays
	QoS Levels (0, 1, 2)	Not mentioned	Level-dependent	Level-dependent
	Delivery Modes (Plaintext, TLS v1.3, TLSMA)	Varies with encryption	Varies with encryption	Varies with encryption
	Payload Size	Not mentioned	Increases communication delay	It affects communication delay

4

PROPOSED QOS MANAGEMENT POLICY

This study utilizes a strong yet flexible methodological approach to boost the energy efficiency of MQTT-enabled IoT networking by implementing machine learning techniques within the decision-making algorithm of the MQTT broker. The methodological consistency approach involves two main investigative streams. The first approach will involve integrating machine learning methods' intense learning into the decision-making algorithm of the MQTT broker. Here, neural networks will be trained using data acquired from a history of topic activity across the network. This training enables the algorithm to adapt to changing network conditions in real-time and to become more accurate in client wake-up timing and energy savings management.

Another studies is conducted is to look at how Quality of Service (QoS) levels can be integrated with the decision algorithm. Machine learning models are applied to enhance the algorithm's capabilities for high-priority message delivery prioritization and energy-efficient wake-up time for subscriber clients. Toward this direction, the method also introduces two operation modes, "idle mode" and "running mode," managing the client behavior efficiently in balance with power consumption and reliability. This elaborate methodology includes using sleep modes, keep-alive interval adjustments, reduction of publication of messages, payload compression, dynamic alteration of QoS levels, and implementation of a decision-making algorithm at the broker's side to boost protocol utility further.

The methodology integrates machine learning to fine tune QoS level prediction models, considering influences from historical data and external factors such as identifying trends, dependencies, and anomalies driving the choice of QoS levels. This calls for developing vital training data sets, the appropriate design of neural architectures, and fine tuning model parameters. The proposed methodology focuses on the comprehensive approach that addresses both the broker and client sides while affirming the importance of detailed power-saving techniques. Potential challenges arising from power-saving techniques, such as increased latency or reduced reliability, are highlighted as following the ultimate balance between energy efficiency and application requirements.

In the final analysis, the MQTT protocol attempts to be applied to powerful and weak gadgets by using techniques incorporating the QoS levels while fitting in "Sleep mode" to reduce latency and enhance reliability. Enhances the performance and sustainability of

MQTT-based IoT networks, giving them an important position as an ingredient for energy efficiency, reliability, and responsiveness in interconnecting ecological network ecosystems.

To evaluate the performance of the proposed algorithm, we conducted a series of experiments using a simulated IoT network with different numbers of publisher and subscriber clients, various QoS levels, and different data patterns. We used the MQTT explorer tool to generate and monitor the MQTT messages. We also used Python programming to implement the machine learning models and the broker's decision-making logic. We compared the energy consumption and reliability of the proposed algorithm with the baseline MQTT protocol and other existing power-saving techniques.

4.1 PROPOSED QOS WORKFLOW

The figure, 1, provides a comprehensive overview of the proposed algorithm, illustrating the efficient interaction between the broker and client sides within MQTT communication, thereby evaluating the performance of IoT devices.

Broker Process: On the broker's side, the following actions are taken:

Data Collection: We cannot find the real data for our thesis, so we are using synthetic data. We are using two datasets: one that contains idle mode, wake-up mode, and probability of failure. Another dataset contains a time stamp, dynamic QoS selection, Static QoS selection, and dynamic and static QoS energy consumption.

Data Analysis: We are utilizing machine learning techniques, including regression, and advanced algorithms like neural networks, such as the MLPRegressor, for data analysis. We are analyzing wake up and idle mode with probability of failure in different network scenarios to predict the QoS levels for dynamic QoS selection.

Decision-Making:

The broker assesses the information received from each client, particularly regarding the reliability of the communication. The goal is to achieve a specified target reliability for the system. The broker evaluates the system's current state and decides whether each client should be active, in sleep mode, or idle. Events, such as wake-up or idle events, influence this decision.

The broker can adjust the QoS levels for individual clients based on the reliability analysis. QoS levels determine the reliability and quality of communication between clients and the broker. The broker considers each client's energy consumption and strives to make decisions that balance communication reliability with energy efficiency. For example, clients may consume different amounts of energy according to their QoS levels and activities.

The broker communicates its decisions to the clients, instructing them to change their QoS levels and enter sleep mode. The decision-making process is dynamic and adapts to changes in the system state. For example, if the system experiences a high-reliability requirement, the broker may instruct clients to operate at higher QoS levels. Conversely, during periods of low activity, clients may be directed to enter sleep mode to conserve energy. The broker continuously evaluates events and adjusts decisions based on environmental conditions. The broker's decision-making process involves monitoring client data and considering reliability, energy consumption, and dynamic adaptation of QoS levels.

Action Execution: When the decision is made, the broker determines the action, such as deciding to Wake up, and then the broker sends a wake-up message to the client, initiating communication. If the decision is to remain in sleep mode, the broker monitors for the next event trigger.

Communication with Client: The broker receives and processes client messages, updating its data for future analysis. Communication is essential for exchanging information between the broker and clients, ensuring synchronization in their operational modes, such as wake-up and sleep mode.

Client Process:

Initialization: The client initiates its operation in either sleep mode or wake-up mode. In sleep mode, active communication is suspended to conserve power; in wake-up mode, the client is ready for communication.

Event Trigger and Wake Up Decision: The broker actively observes the status of each client, identifying instances where a client is detected as inactive for an extended period. Upon detecting such inactivity, the broker initiates a decision-making process to optimize energy consumption. Consequently, the broker instructs the inactive client to enter sleep mode, ensuring efficient resource utilization during periods of inactivity. In sleep mode, the interrupt controller remains active, which tends to receive interrupts from the environment. The environment interrupts can take various forms, such as priority tasks, external requests, or critical events.

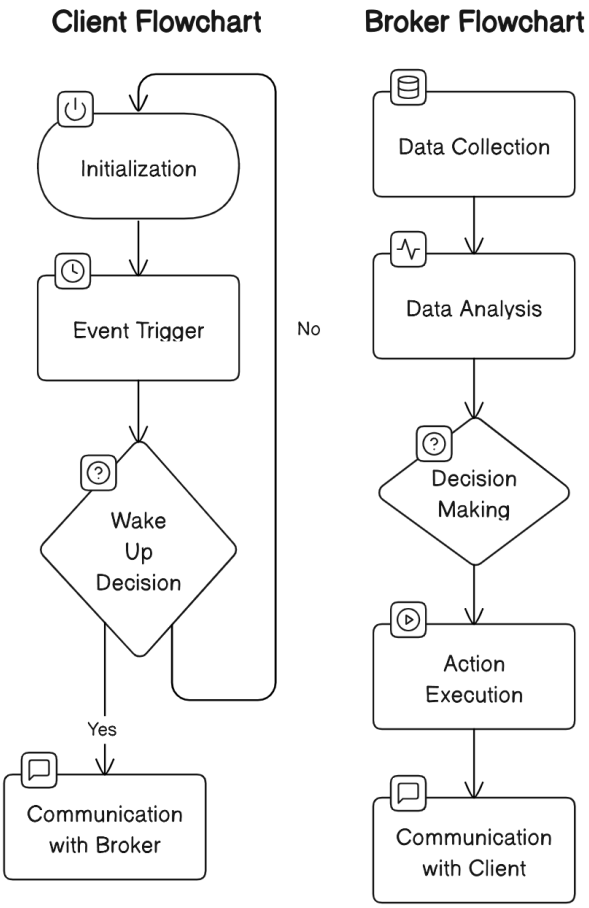
An external event occurs when an interrupt is transmitted to the inactive client. This interrupt acts as a wake-up call, requesting the client change from sleep mode to an active state. Following the awakening, the client negotiates with the broker to determine the appropriate Quality of Service level. This negotiation aims to align the client's operational parameters, including communication reliability and energy efficiency, within the environmental conditions.

Communication: If the decision is to wake up, the client enters an active communication state. Establishes communication with the broker, potentially transmitting or receiving data. Following the commu-

nication process, the client may reset timers or reevaluate conditions before deciding to go back to sleep.

The client will return to a low-power state if it stays in sleep mode. For improved adaptability and responsiveness in IoT communication.

Figure 1: Proposed QoS Workflow



4.2 EXPERIMENTAL SETUP

In our research work, we have thoroughly evaluated the low-power IoT messaging protocol LP-MQTT to determine its effectiveness in managing energy consumption. The evaluation methodology has been based on an experimental design implemented using three strategies. Statical dynamic QoS, MLPRegressor model strategy, and static QoS. These strategies were strategically applied in the MQTT-enabled IoT connection scenario, where a message broker and several clients interacted via a publish-subscribe protocol.

4.2.1 *Static QoS Strategy*

The Static QoS strategy had predetermined QoS levels for specific events initiated instead of dynamic strategies. The QoS levels are set before the simulation process starts and remain unaltered during the entire simulation run. The static QoS strategy was used as a baseline of comparison to determine the extent of the impacts dynamic QoS adjustments have on energy consumption in LP-MQTT-enabled IoT systems. Static QoS levels provided a benchmark below, which could be used to evaluate the dynamic logically.

4.2.2 *Statical Dynamic QoS Strategy*

On the other hand, machine learning techniques were used to make dynamic predictions of QoS levels from the past in the Statical Dynamic QoS strategy. This model has been trained to analyze historical data sets, allowing real-time adaptation of the developed network control time to the LP-MQTT protocol. In this way, it was thought that dynamically adjusting the QoS levels of a protocol would improve the overall efficiency and responsiveness of the LP-MQTT protocol by allowing it to adapt to the changing nature of the network environment.

4.2.3 *MLPRegressor Model Strategy*

The MLPRegressor model was essential for the static dynamic quality of the service strategy. This machine learning model experienced training on historical datasets to dynamically predict QoS levels. The MLPRegressor dynamically adjusted QoS levels as network conditions evolved to optimize the LP-MQTT protocol's performance. The model's training process involved learning patterns from past data, enabling it to make informed decisions about the appropriate level of QoS in response to the current conditions. This dynamic adaptation balanced the QoS requirements and energy efficiency of the IoT system.

We performed accurate experiments to investigate complex trade-offs between different quality of service (QoS) regulations and how they affect the energy efficiency of Internet of Things (IoT) devices. Using LP-MQTT and modifications to the MQTT protocol as examples, we use these strategies to deal with issues related to MQTT-capable Internet of Things clients subject to more extended battery life restrictions. Our results, obtained via simulations using Python code in various settings, provide a new perspective on maximizing the performance efficiency of Internet of Things systems under various dynamic network situations.

4.2.4 *Simulation Step Description*

Figure 2 presents a comprehensive visualization of the simulation step, an integral component in evaluating Quality of Service (QoS) strategies within LP-MQTT-enabled IoT systems. This simulation process is designed to assess the efficacy of different QoS policies under diverse network conditions, providing insights into their impact on IoT device performance and energy efficiency.

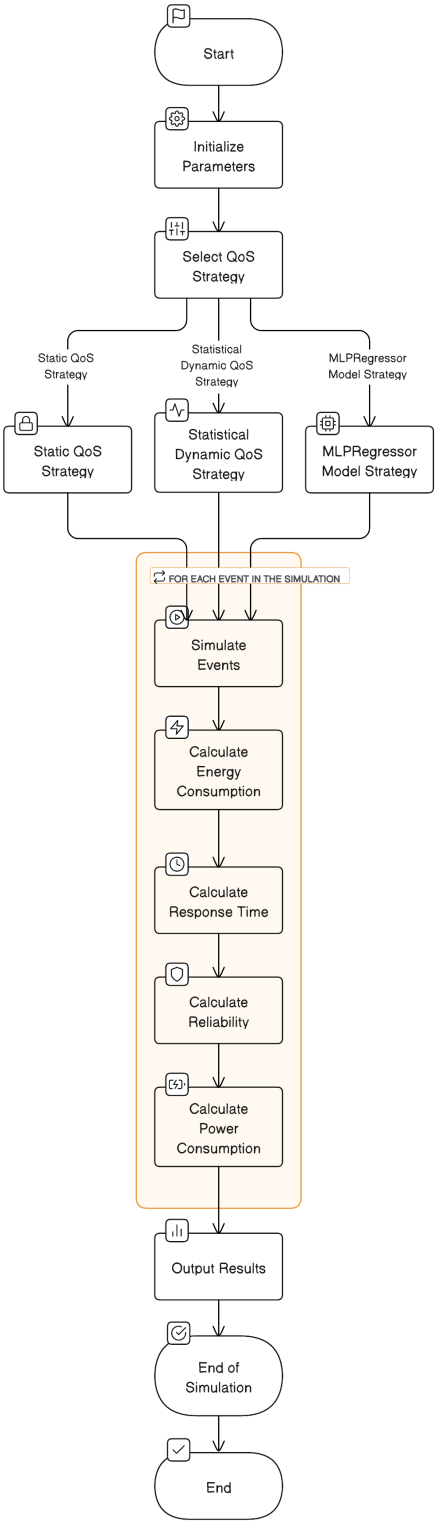
The simulation commences by initializing crucial parameters, such as the failure probability distribution and energy consumption coefficients. Introducing a stochastic element in the simulation mirrors the inherent uncertainty in real-world scenarios. This randomness is allied to a coin flip generated during the decision-making phase, contributing to the simulation's adaptability to unforeseen variations.

Within the simulation, we were chosen among three distinct QoS strategies: Static, Statical Dynamic, and MLPRegressor Model. The Static QoS Strategy adheres to pre-determined QoS levels, offering a baseline comparison. In contrast, the statical dynamic and MLPRegressor model dynamically adjust QoS levels based on historical data and machine learning, introducing an element of learning and adaptation.

As the simulation progresses through events, the stochastic mechanism comes into play, simulating the randomness inherent in dynamic network conditions. This includes the generation of messages, similar to a coin flip by random generation, influencing the decision-making process within each iteration. The stochastic nature ensures that the simulation captures the variability present in real-world IoT environments.

Throughout the iterative process, the simulation calculates crucial metrics such as energy consumption, response time, reliability, and power consumption based on the chosen QoS strategy. The results serve as valuable output for in-depth analysis, facilitating a comprehensive understanding of the trade-offs associated with each QoS policy. This comprehensive approach, incorporating deterministic and stochastic elements, enriches the simulation's capability to mimic real-world scenarios and aids in making informed decisions for optimizing IoT device performance and energy efficiency.

Figure 2: Simulation Step



4.2.4.1 *Simulation Parameters:*

- **Failure Probability Distribution:** Normal distribution with mean (μ) and deviation (σ).
- **Power Consumption Components:**
  - Em: Energy consumption for sending a message
  - Ecomp: Energy consumption for computing the QoS level
  - Esw: Energy consumption for switching the QoS level
  - Eidle: Energy consumption for being idle
  - Eton: Energy consumption for turning on the device
  - Etof: Energy consumption for turning off the device
- Energy Consumption Coefficients (Em, Ecomp, Esw, Eidle, Eton, Etof): Parameters influencing energy consumption based on event type and QoS level.
- Alpha Value α: Constant value of 100.
- **Energy Consumption Formula:**

  $energy\_consumption = Em + Ecomp + Esw + Eidle + Eton + Etof$
- **Response Time Formula:**

  $response\_time = current\_timeâtimestamp$
- **Reliability Calculation:**

  $R = 1 - pq$

  Where p is the packet failure probability, and q is the number of transmissions.

  For QoS levels:
  - QoS 0 : $q = 1$, so $R = 1 - p$
  - QoS 1 : $q = 2$, so $R = 1 - p^2$
  - QoS 2 : $q = \infty$, so $R = 1$
- **Power Consumption Calculation:**

  $PowerConsumption = EnergyConsumption/Time$

These parameters and formulas were utilized to comprehensively evaluate the performance of different QoS strategies under varying network failure scenarios.

4.2.5 *Assumptions Related to LP-MQTT Protocol Implementation*

4.2.5.1 *Reliability of LP-MQTT Protocol*

If the communication channels and network conditions of the LP-MQTT low-power IoT messaging protocol fit a level of reliability presented in a typical low-power IoT environment. This includes assuming that message delivery, acknowledgments, and QoS level adjustment will be reliable without significant packet disruptions or loss. For example, any unanticipated irregularities in communications' reliability will impact the observed performance for the LP-MQTT protocol, and the nature of typical network behavior is considered in the interpretation of results.

4.2.5.2 *Stability of LP-MQTT Protocol*

We assume the LP-MQTT low-power IoT messaging protocol operates stably throughout the simulation. This implies that the enhancements to the protocol concerning new machine learning techniques and providing dynamic QoS adjustments do not introduce any unseen instabilities or disruptions in the communication between MQTT message brokers and clients.

4.2.5.3 *Consistency of Machine Learning Predictions*

We supposed that every prediction made at each time with the machine learning model (MLPRegressor) implemented in the Statical Dynamic QoS strategy is accurate concerning historical measurements. Considering that predicting observations' quality defines, to a great extent, either successful or failed adaptation of the dynamic QoS, we could fulfill this aspect within the scope of our assumptions analysis. Any fluctuations or imperfections may affect the performance of the LP-MQTT protocol regarding these machine-learning predictions.

4.2.5.4 *Energy Consumption Model Accuracy*

It is assumed that the model for energy consumption, especially given the enhancements of the LP-MQTT protocol, is accurate. In evaluating impacts that will be felt on the IoT device's energy efficiency, the energy consumption model influenced by parameters like QoS levels and dynamic adjustments has relevance. Any discrepancies or limitations in this model's accuracy could automatically affect energy consumption evaluations.

4.2.6 *Assumptions Related to Experimental Design and Simulation Parameters*

4.2.6.1 *Representative Schedule Data*

We would assume that this schedule data, as used in simulation, events, and corresponding QoS levels, represents real-world scenarios. The effectiveness of the LP-MQTT protocol and the QoS approach stands as long as the extent of all the categorized events to be covered within the given schedule is relatively relevant and diverse. The conclusions from the simulation work might have to be based on the deviations from realistic scenarios.

4.2.6.2 *Consistent Network Conditions*

The simulation assumes consistency in the network conditions kept constant throughout each cycle. However, it does not model explicit variations with message delivery times or issues concerned with connectivity. The relatively stable network assumption conditions ensure the focus on how QoS adjustments affect the responses of the LP-MQTT protocol.

4.2.6.3 *Uniform IoT Device Characteristics*

When participating in the MQTT-enabled connections, we assume similarity in the characteristics of the IoT devices in question. This assumption allows us to take consistent hardware specifications for granted, energy consumption patterns, and responsiveness to QoS changes. The similarity in the device characteristics could introduce additional variables whose impact on the validity of the simulation results would have to be studied separately.

4.2.6.4 *Applicability of LP-MQTT Enhancements*

On the other hand, adjusting dynamic QoS and low-power messaging protocols are considered effective enough to be applied to a wide range of IoT applications. The scenario-specific insights gained from the simulation make it challenging to generalize the findings for other IoT use cases, so this is only possible if the general applicability of LP-MQTT improvements can be assumed.

These assumptions help frame the context in which the evaluation of the LP-MQTT protocol takes place and according to which the experimental parameters will be defined. It is mandatory that these assumptions are taken into account while the interpretation of the results given by the simulations is followed and the negative or positive assessments made regarding the performance of the LP-MQTT protocol in terms of energy efficiency under different strategies.

5

RESULTS

In this study, we examined the performance of power management strategies within the context of LP_MQTT (Low Power MQTT). The primary focus was understanding how these strategies respond to scenarios characterized by varying probabilities of network failure. We employed a probabilistic model to simulate network failures, assuming a normal distribution to mimic real-world environmental changes.

The probability of network failure serves as a crucial parameter in evaluating the robustness and adaptability of LP_MQTT in dynamic and unpredictable environments. We chose a normal distribution due to its versatility in representing a wide range of scenarios, and we utilized the mean and deviation to control the characteristics of the simulated network conditions.

- High Failure Scenarios (mean = 0.5): In these scenarios, we set the mean probability of network failure to 0.5, indicating a higher likelihood of significant environmental changes. The higher deviation associated with these scenarios reflects a harsher and more unpredictable environment. We aimed to assess how power management strategies perform when faced with substantial and frequent network disruptions.

- Medium Failure Scenarios (mean = 0.25): The medium failure scenarios were designed to test the adaptability and stability of LP_MQTT. With a mean probability of failure set at 0.25, these scenarios represented moderate environmental changes. The goal was to assess how well the power management strategies balance conserving energy during periods of stability and quickly responding to network failures when they occur.

- Low Failure Scenarios (mean = 0.05): For scenarios with a mean probability of failure set at 0.05, we simulated a stable environment with lower intensity fluctuations. These scenarios aimed to represent a more predictable and less harsh operational environment. By examining the performance of LP_MQTT under such conditions, we gained insight into how the system behaves when network failures are infrequent and less disruptive.

The results highlight the adaptability and potential energy efficiency of dynamic QoS strategies in LP_MQTT, especially in scenarios with higher probabilities of network failure, where fluctuations in environmental conditions are more pronounced. The parameters and formulas introduced contribute to a comprehensive understanding

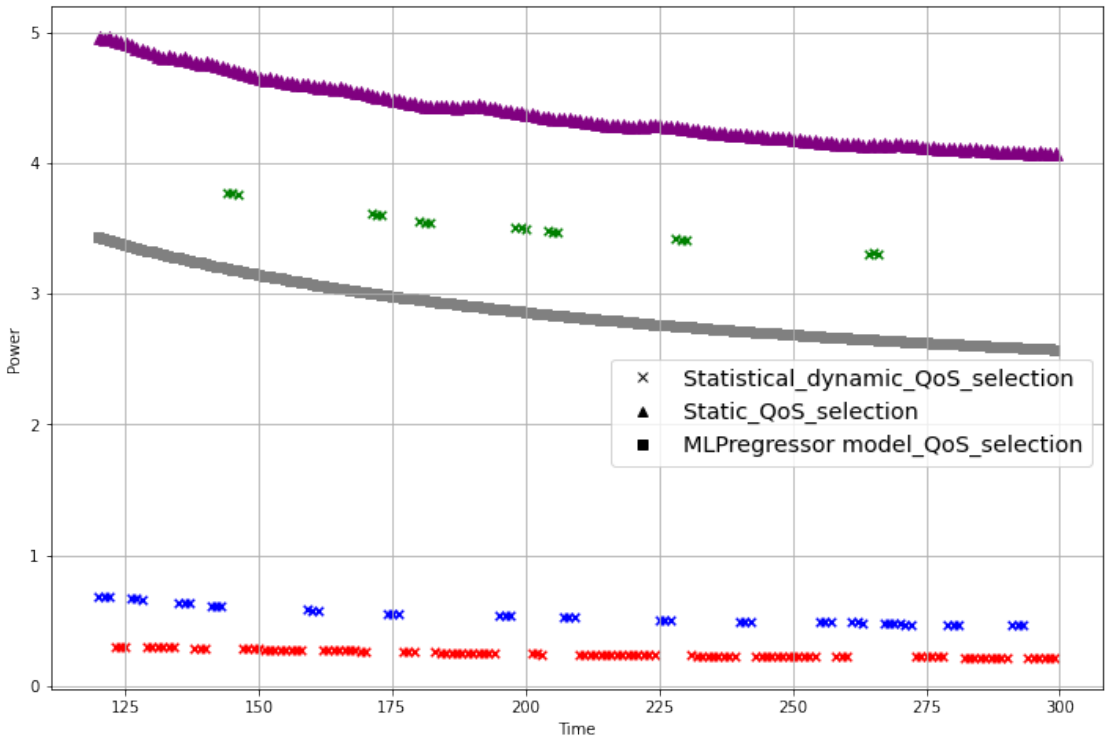
of the simulation experiments and their implications for the power management of LP_MQTT.

5.1 POWER CONSUMPTION ACROSS DIFFERENT STRATEGIES

The graphs (as indicated in the graphs Figure 3, Figure 4, Figure 6, Figure 5, Figure 8, and Figure 7,) provides a comprehensive view of the power consumption dynamics related to different Quality-of-Service (QoS) strategies over time.

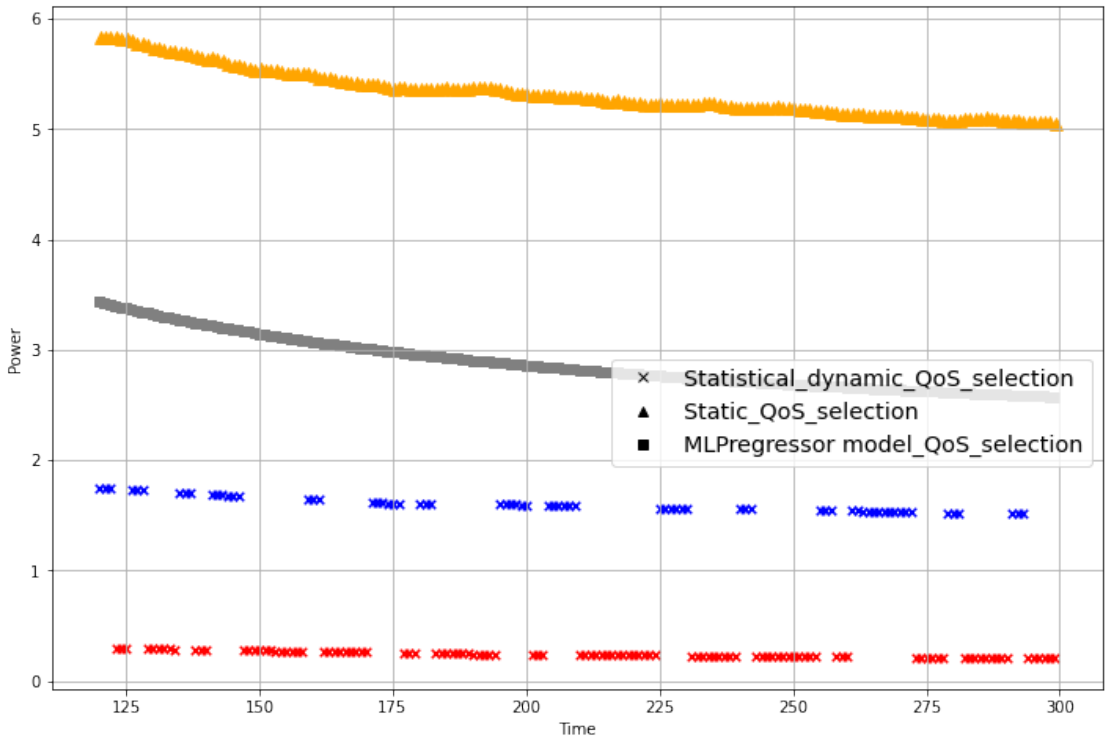
Figure 3 depicts three distinct Quality of Service (QoS) strategies: Statical Dynamic QoS, Static QoS, and MLPregressor model QoS. A specific symbol represents each strategy - Statical Dynamic QoS is denoted by a cross, Static QoS by a triangle, and MLPregressor model QoS by a square. The color-coded representation indicates the selected QoS level for each strategy. The Static QoS strategy, symbolized by purple triangles, maintains a fixed QoS level (QoS 2) throughout the evaluation. On the other hand, the MLPregressor model QoS strategy, illustrated with gray squares, also used for a constant QoS level (QoS 2). Statical Dynamic QoS, by contrast, can adapt dynamically to changes in the environment. This dynamic adaptation is visually presented with red colour for QoS 0, blue for QoS 1, and green for QoS 2. Despite its dynamic nature, the Statical Dynamic QoS strategy consistently achieves lower power consumption than the Static QoS strategy. Statical Dynamic QoS's color-coded QoS levels illustrate the strategy's efficiency in optimizing power consumption under changing environmental conditions. The comparative analysis highlights that the Statical Dynamic QoS strategy consistently achieves lower power consumption, showcasing its enhancing capabilities under varying environmental conditions.

Figure 3: Power Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.5,Std=0.1)



In Figure 4, the static QoS strategy, indicated by orange triangle points, consistently favors QoS 1. Statical dynamic QoS fluctuates between QoS 0 (red) and 1 (blue), while the MLPRegressor continuously corresponds to QoS 2, represented by gray points. The power consumption patterns among these strategies differ significantly. The static QoS strategy exhibits high power consumption with its persistent choice of QoS 1 (orange points). In contrast, the Statical Dynamic QoS strategy, particularly when opting for QoS 0 and 1, and the MLPRegressor strategy, consistently favoring QoS 2, demonstrate lower power consumption. Interestingly, despite the Static QoS strategy's tendency to favor QoS 1, the MLPRegressor strategy achieves relatively lower power consumption, supporting the efficiency of the MLPRegressor approach in minimizing power usage at a higher QoS level than the Static QoS strategy.

Figure 4: Power Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.5,Std=0.01)



In Figures 6 and 5, depicting scenarios with a medium failure rate, the chosen Quality of Service (QoS) strategies exhibit distinctive power consumption trends. In Figure 5, the static QoS strategy consistently selects QoS 1(orange), resulting in relatively higher power consumption. In contrast, the Statical Dynamic QoS strategy dynamically adjusts between QoS 0(red) and 1(blue) based on the probability of failure, achieving comparatively lower power consumption. Simultaneously, the MLPRegressor Model Strategy diverges by selecting QoS 2(gray). In Figure 6, the Static QoS strategy selects QoS 0(brown), whereas the Statical Dynamic QoS strategy also selects QoS 0(red), showcasing lower power consumption. QoS 2(gray) continues to be chosen by the MLPRegressor Strategy. These figures illustrate the varied power consumption dynamics across strategies under medium failure rates, with Statical Dynamic consistently exhibiting lower power consumption than Static.

Figure 5: Power Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.25,Std=0.05)

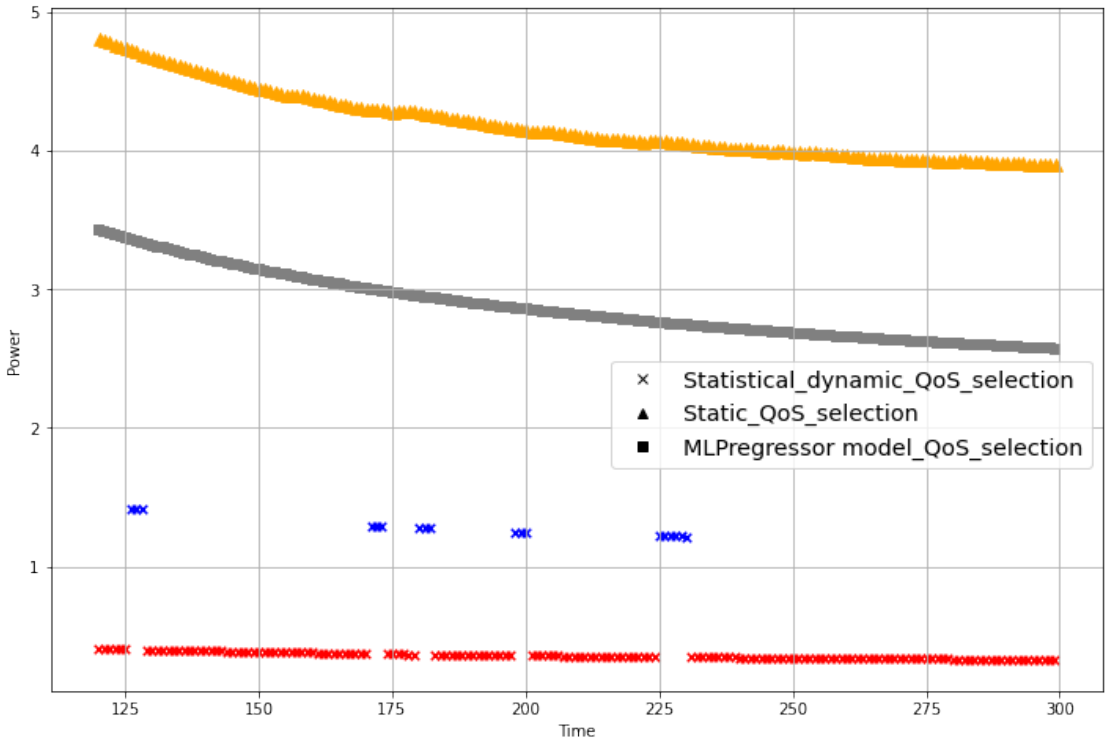
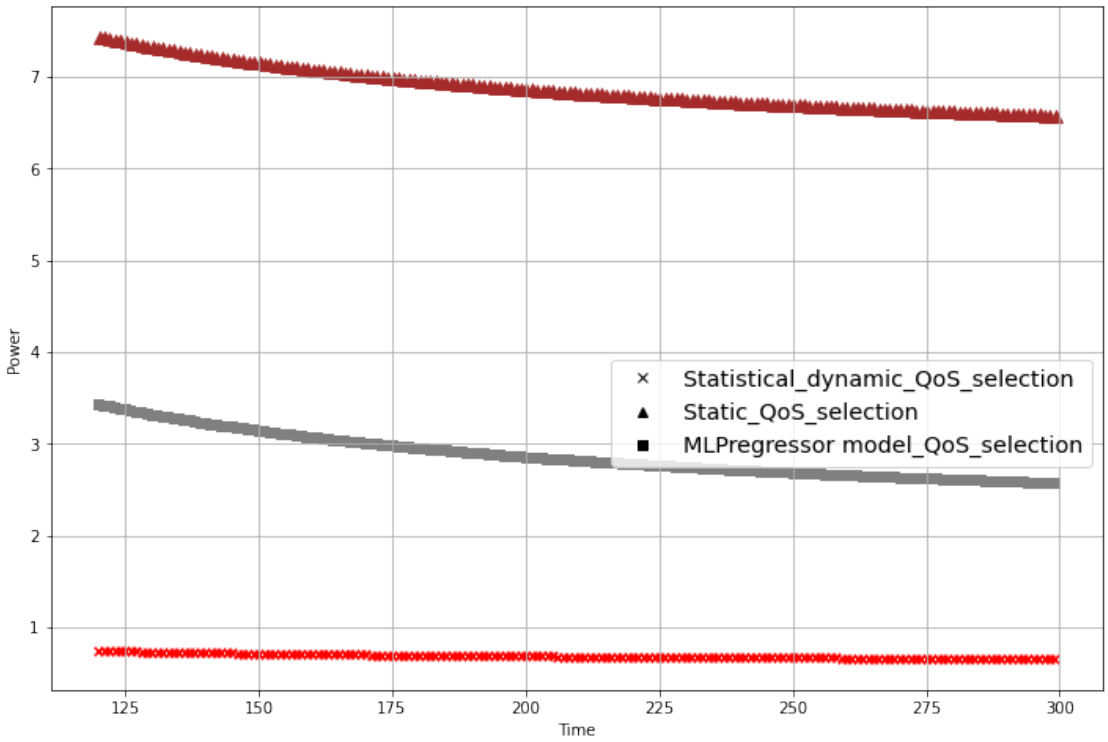


Figure 6: Power Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.25,Std=0.005)



In Figures 8 and 7, representing scenarios with low failure rates and distinct quality-of-service (QoS) strategies exhibit varied power consumption behaviors. The Static QoS strategy consistently select for QoS 0(brown), resulting in higher power consumption levels, while the Statical Dynamic QoS strategy dynamically adapts and selects QoS 0(red), achieving lower power consumption. The MLPRegressor Model Strategy diverges in both instances by selecting QoS 2(gray). The MLPRegressor model prioritizes a different QoS level from the static and Statical dynamic strategies.

Figure 7: Power Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.05,Std=0.01)

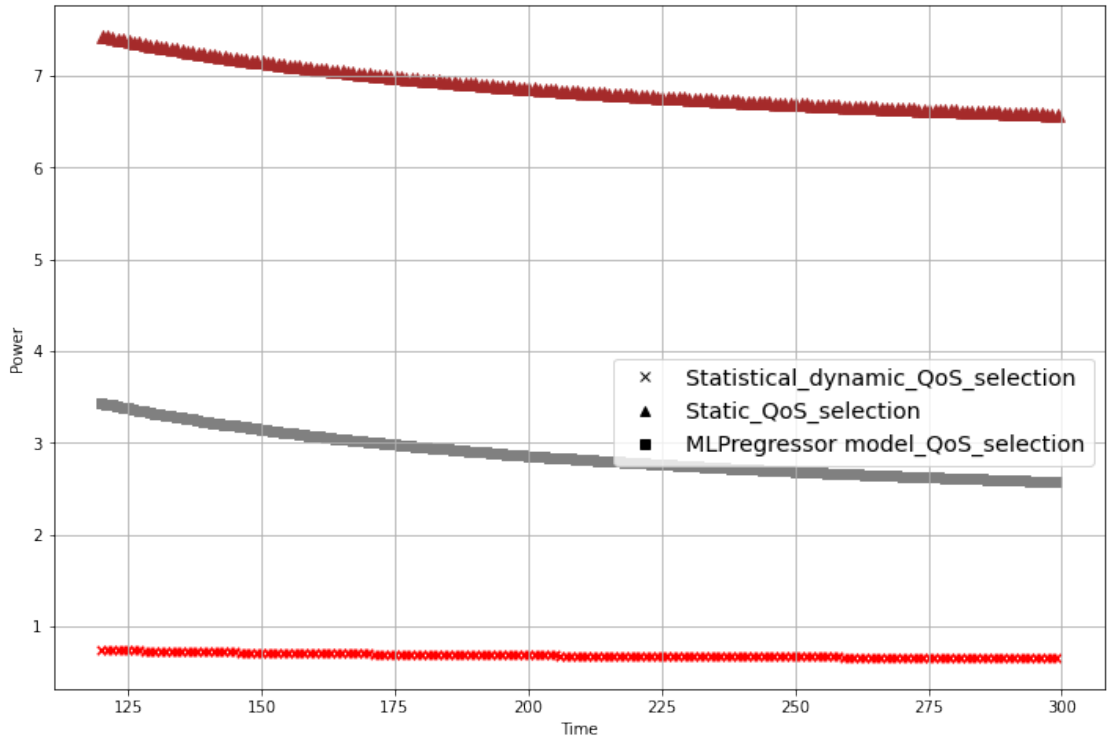
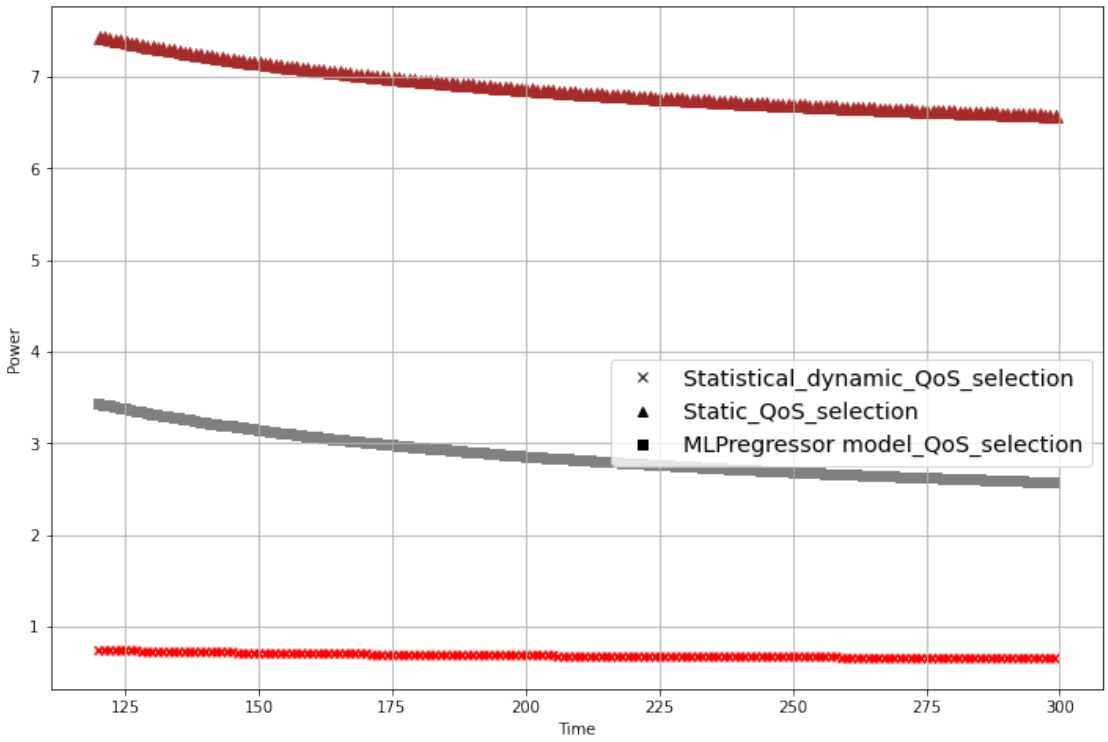


Figure 8: Power Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.05,Std=0.001)



The graphs show that power consumption is highest in the static QoS approach, lowest in the statical dynamic QoS approach, and in the MLPregressor model, it is between static and statical dynamic QoS. Based on the graphs, we conclude that the statical dynamic QoS model has a low power consumption.

5.2 RELIABILITY ACROSS DIFFERENT STRATEGIES

Graphs Figure 9, Figure 10, Figure 11, Figure 12, Figure 13, and, Figure 14 present the reliability analysis results showing the performance of each strategy running under different probabilities of failure. In situations with a high likelihood of failure, the Statical dynamic QoS proved adaptable to the challenge, remaining highly reliable compared to the MLPRegressor model strategy and static QoS. In nature, Statical dynamic QoS is adaptive; hence, it can capture some difficult network conditions that would lower its reliability.

Figure 9: Relibility Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.5,Std=0.01)

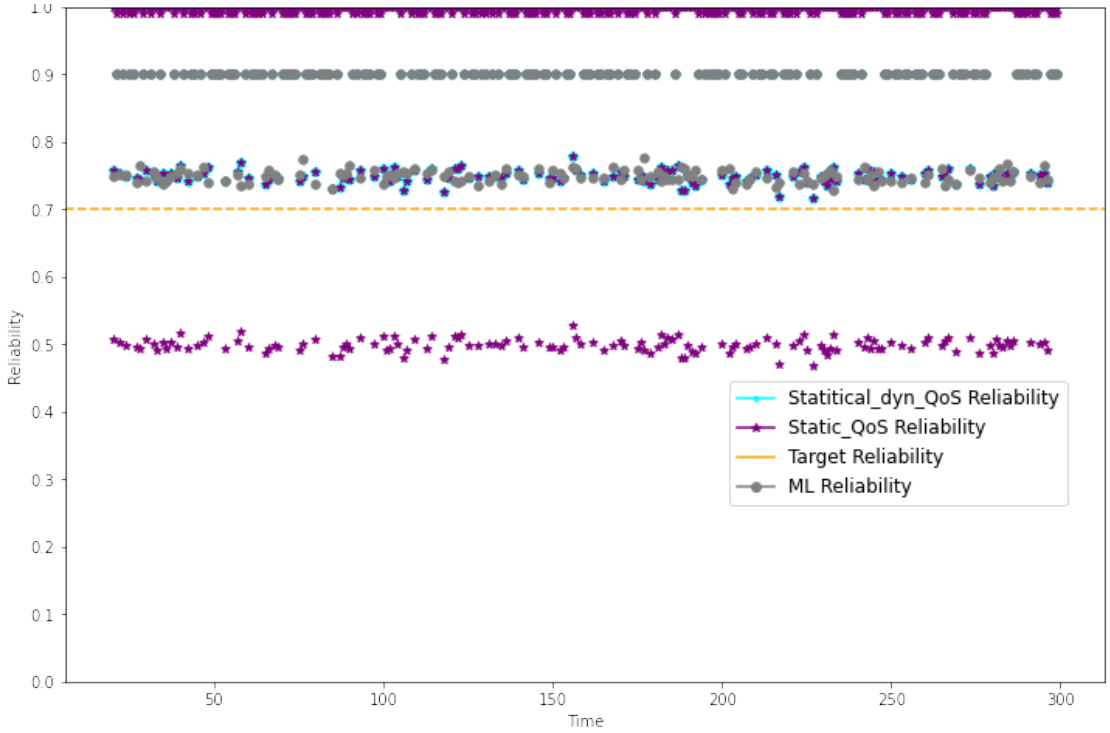
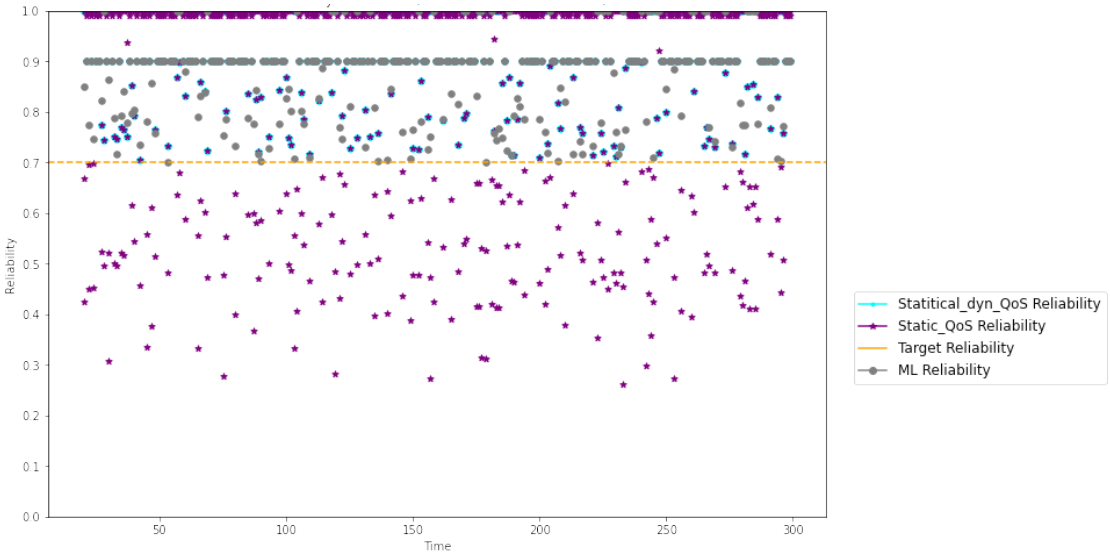


Figure 10: Relibility Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.5,Std=0.1)



In Figures 9 and 10, depicting scenarios characterized by a high probability of failure, the focus is on the reliability of different Quality of Service (QoS) strategies. The x-axis represents time in seconds, while the y-axis indicates reliability. In particular, the target reliability is set at 0.7(yellow line). In Figure 9, the static QoS (purple

color) strategy occasionally drops below the target reliability, reflecting periods of reduced performance. In contrast, the statical Dynamic QoS(cyan color) and MLPRegressor Model(gray color) strategies consistently surpass the target reliability, demonstrating their robust adaptability and superior performance under conditions of heightened failure probability. Figure 10 supports this trend, with the statical Dynamic and MLPRegressor strategies consistently maintaining reliability above the set target. In contrast, the static strategy encounters discontinuous challenges in meeting the specified reliability threshold. These results underscore the enhanced reliability of the dynamic adaptability of statical Dynamic and MLPRegressor strategies compared to the more rigid Static approach.

Figure 11: Relibility Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.25,Std=0.005)

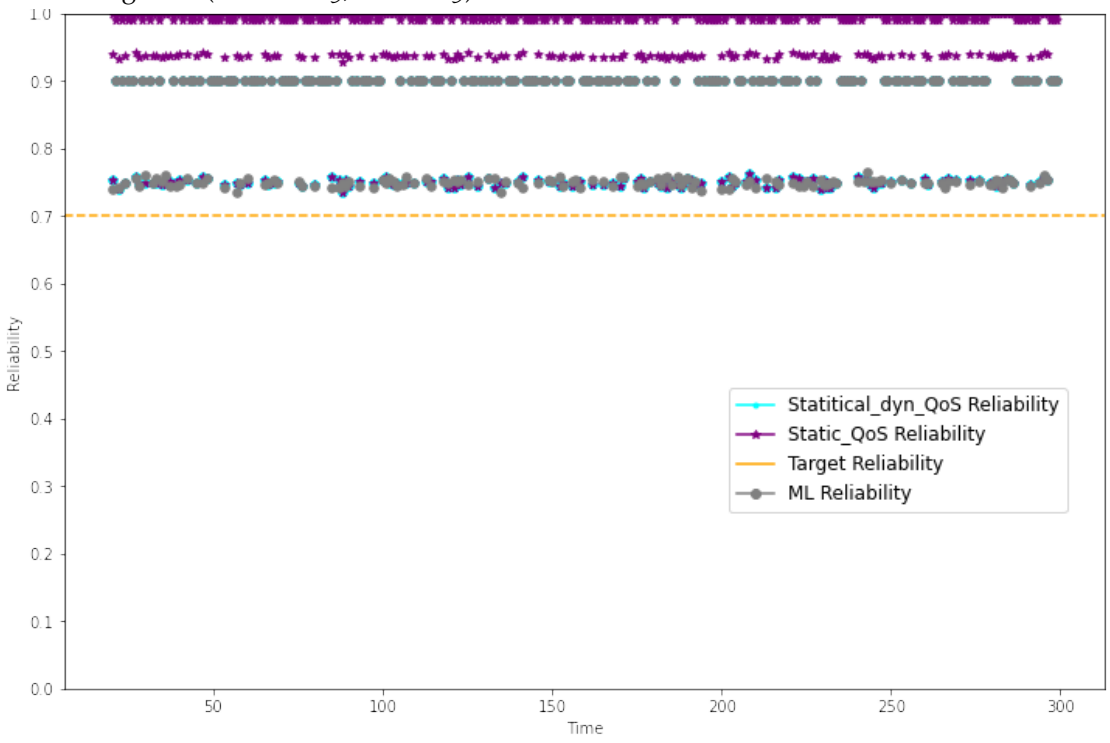
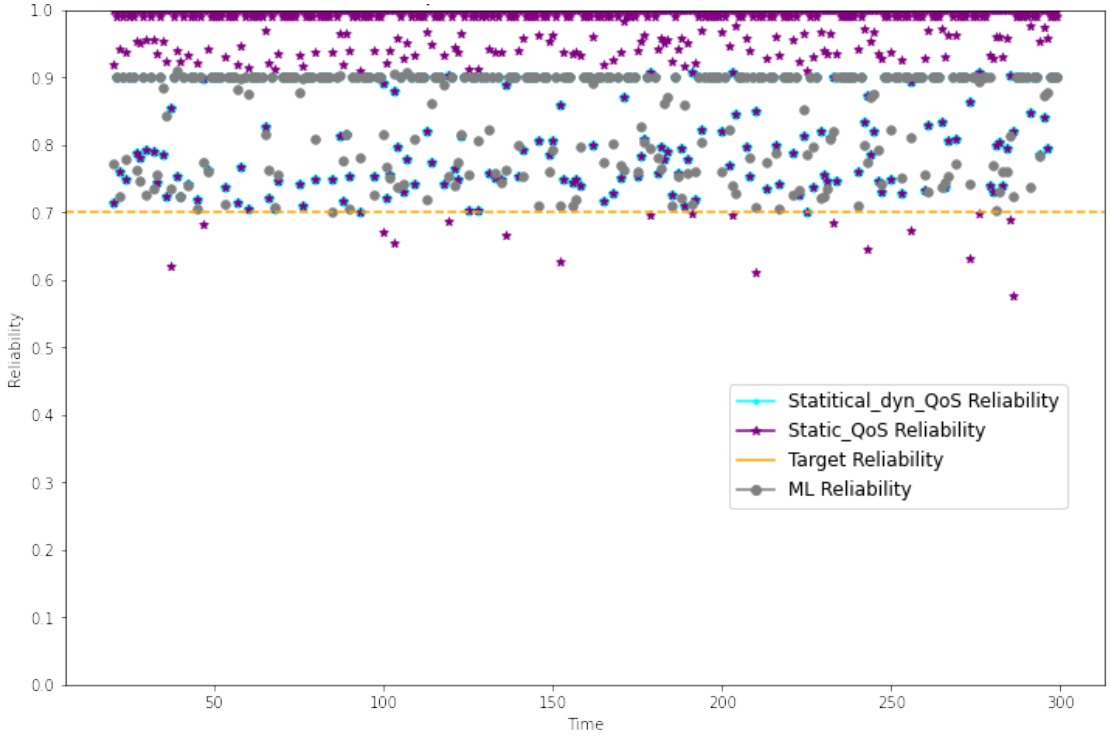


Figure 12: Relibility Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.25,Std=0.05)



Figures 11 and 12 depict scenarios characterized by a medium probability of failure; the primary focus is the reliability of various Quality of Service (QoS) strategies. The x-axis represents the time in seconds, while the y-axis signifies the reliability. In Figure 11, all strategies consistently maintain reliability levels above the specified target throughout the observed time, indicating robust performance despite moderate failure probabilities. However, in Figure 12, the Static QoS strategy experiences occasional dips below the target reliability, suggesting periods of decreased performance. In contrast, the statical Dynamic QoS and MLPRegressor Model strategies consistently remain above the target reliability, showcasing their superior adaptability and reliability under conditions of moderate failure probability. These findings emphasize the reliability of statical dynamic and MLPRegressor strategies compared to the occasional reliability challenges encountered by the more rigid static approach in scenarios with medium failure probability.

Figure 13: Relibility Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.05,Std=0.001)

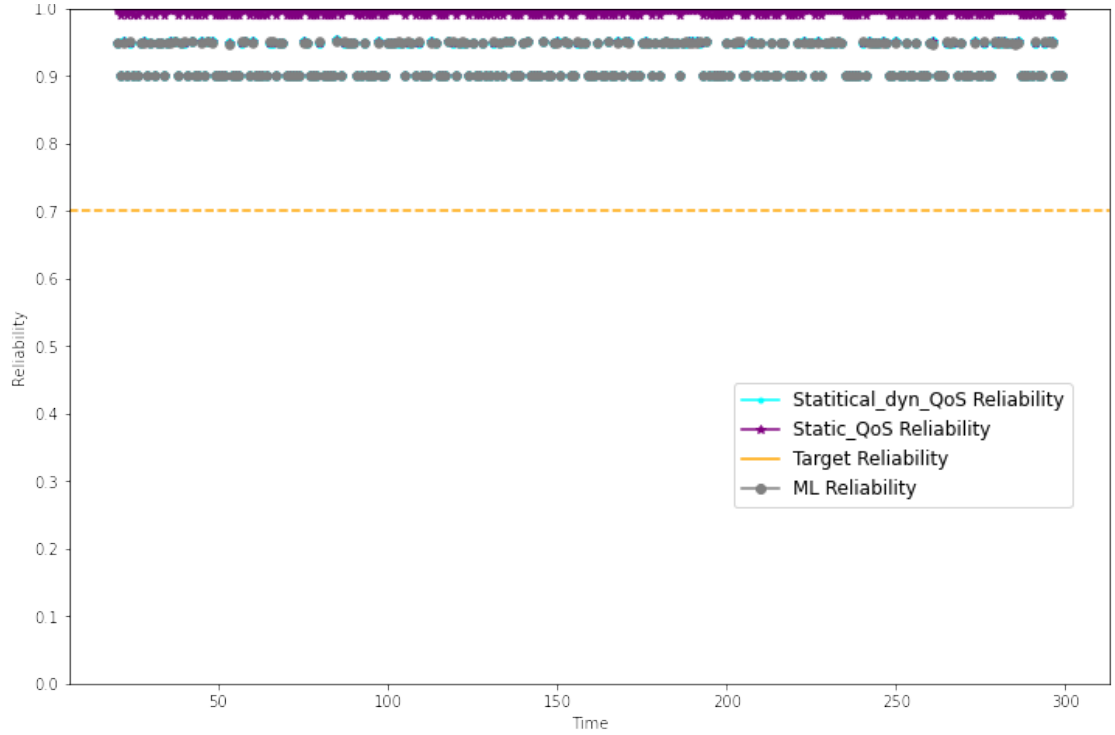
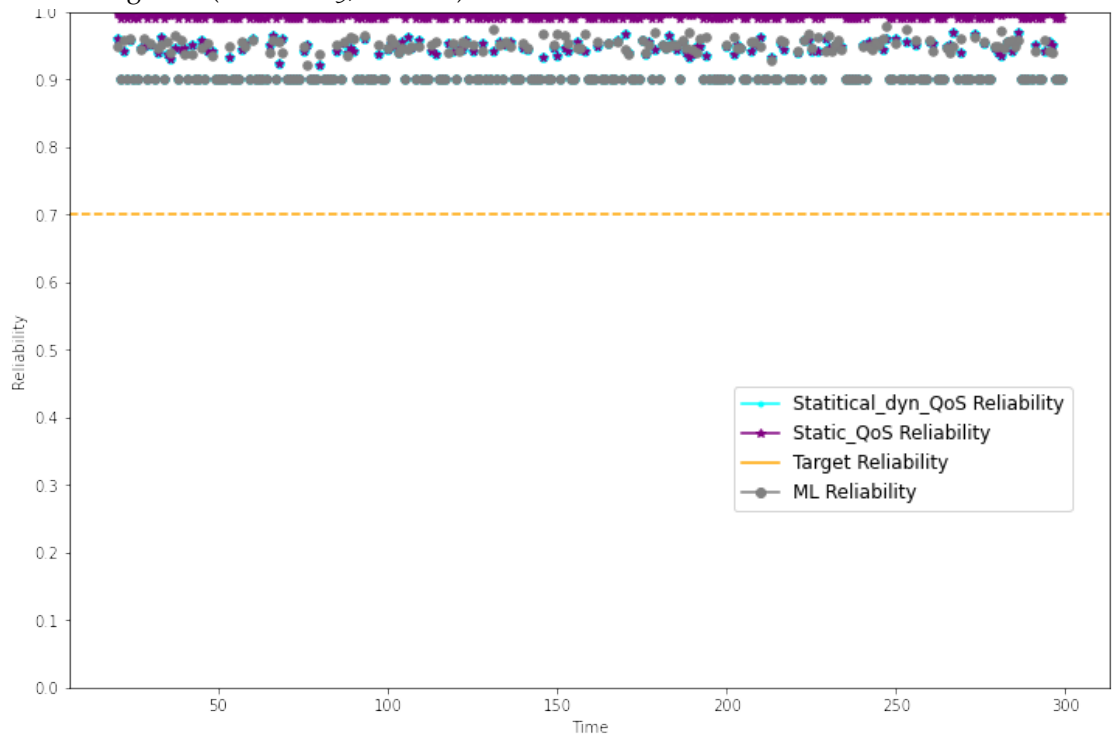


Figure 14: Relibility Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.05,Std=0.01)



In Figures 13, and 14, illustrating scenarios characterized by a low probability of failure, the focus is on the reliability of various Quality-of-Service (QoS) strategies. The x-axis represents time in seconds, while the y-axis indicates reliability. All strategies consistently maintain reliability levels above the set target throughout the observed time. In both figures, surrounding low failure probability situations, the Static, Statical Dynamic, and MLPRegressor Model strategies consistently exceed the specified target reliability, showcasing their powerful performance in ensuring a high level of reliability in the face of minimal failure events. These results indicate the effectiveness of all three strategies in maintaining reliable performance under conditions of low failure probability. The figures represent a stable and reliable operation throughout the observed time duration, highlighting the flexibility of the selected QoS strategies.

We can conclude from the graphs above that our proposed models are more reliable than the static QoS approach since we consistently achieve the target reliability. In contrast, the static QoS approach does not consistently achieve the target reliability.

5.3 RESPONSE TIME COMPARISON

The response time analysis resolve the efficiency of every strategy in message delivery under the LP-MQTT protocol. The high probability of failure scenarios introduced increased response times in all strategies, with Statical Dynamic QoS showcasing its adaptiveness by minimizing delays. Graphs Figure 15, Figure 16,Figure 17,Figure 18 Figure 19, and Figure 20, present the response time results.

Figures 15 and 16, depicting scenarios characterized by a high probability of failure, response times of different Quality of Service (QoS) strategies are compared. The x-axis represents time in seconds, while the y-axis signifies response time. The MLPRegressor strategy (green line) exhibits a high response time, indicating potentially longer processing delays. In the Statical Dynamic (blue line) strategy, the response time is moderate, whereas in the Static (yellow line) strategy, the response time is low, suggesting faster event processing.

Figure 15: Response time Vs Time Using Static, Statical_dynamic and ML-PRegresser(Mean=0.5,Std=0.01)

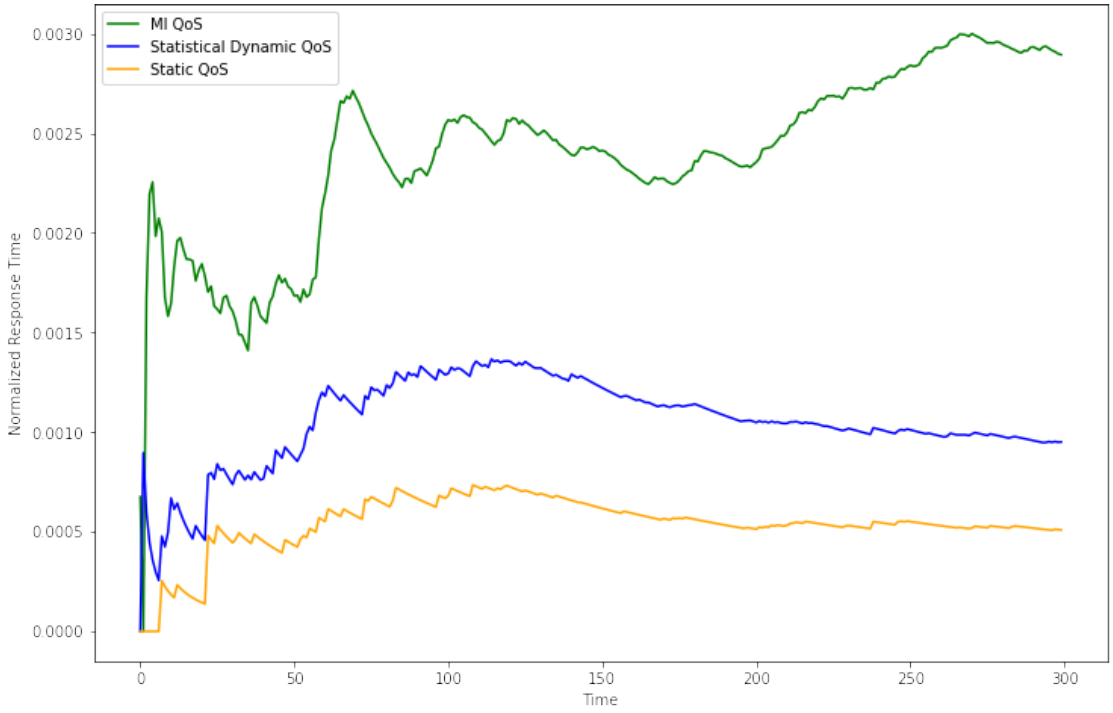
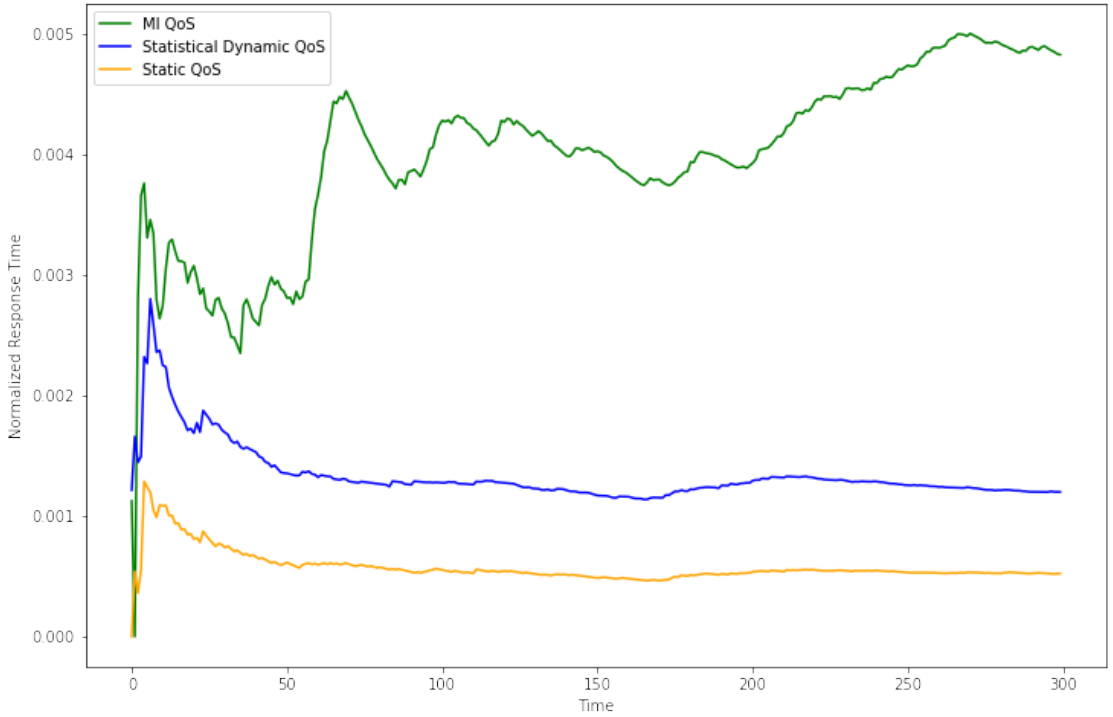


Figure 16: Response time Vs Time Using Static, Statical_dynamic and ML-PRegresser(Mean=0.5,Std=0.1)



Similarly, in Figures 17 and 18, also representing the low probability of failure scenarios, the MLPRegressor strategy continues to exhibit a high response time. The Statical Dynamic strategy maintains a moderate response time, and the Static strategy encounters a decrease in response time. These observations suggest that, under conditions of low failure probability, the MLPRegressor strategy consistently incurs longer response times compared to the more responsive Static and moderately responsive Statical Dynamic strategies.

Figure 17: Response time Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.25,Std=0.005)

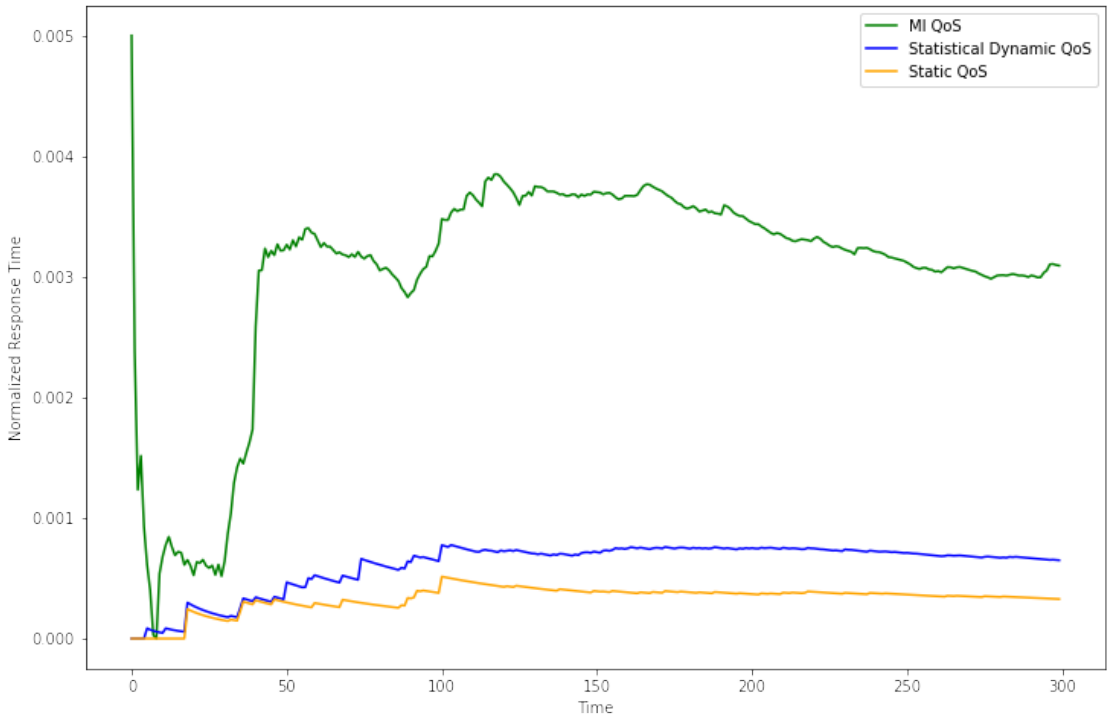
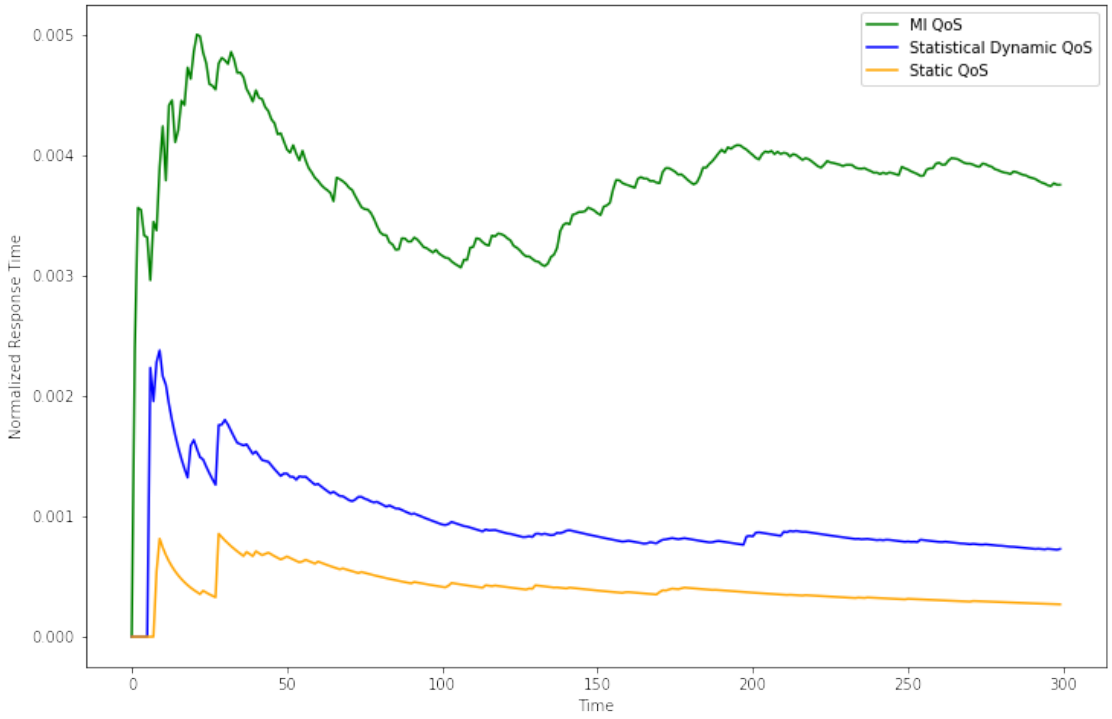


Figure 18: Response time Vs Time Using Static, Statical_dynamic and MLPRegresser(Mean=0.25,Std=0.05)



Furthermore, Figures 19 and 20 show scenarios with a low probability of failure. The MLPRegressor strategy maintains a high response time, indicating potential delays in event processing. The Statical Dynamic strategy exhibits a moderate response time, while the Static strategy encounters a decrease in response time, reflecting more efficient event processing during periods of low failure probability.

Figure 19: Response time Vs Time Using Static, Statical_dynamic and ML-PRegresser(Mean=0.05,Std=0.001)

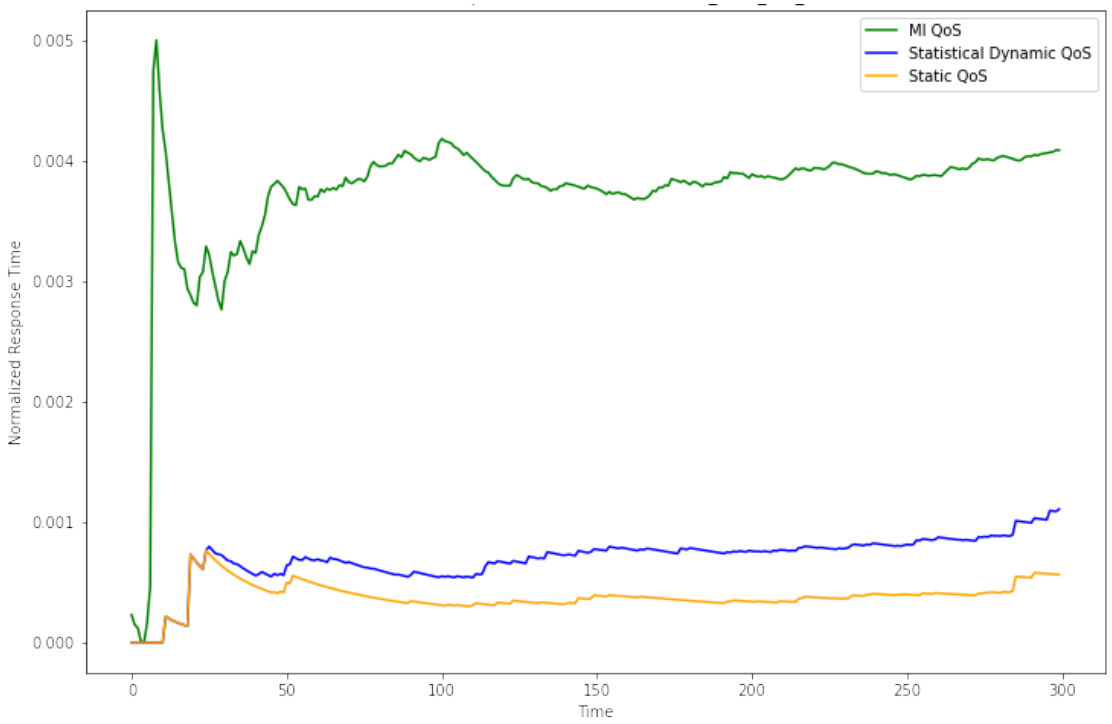
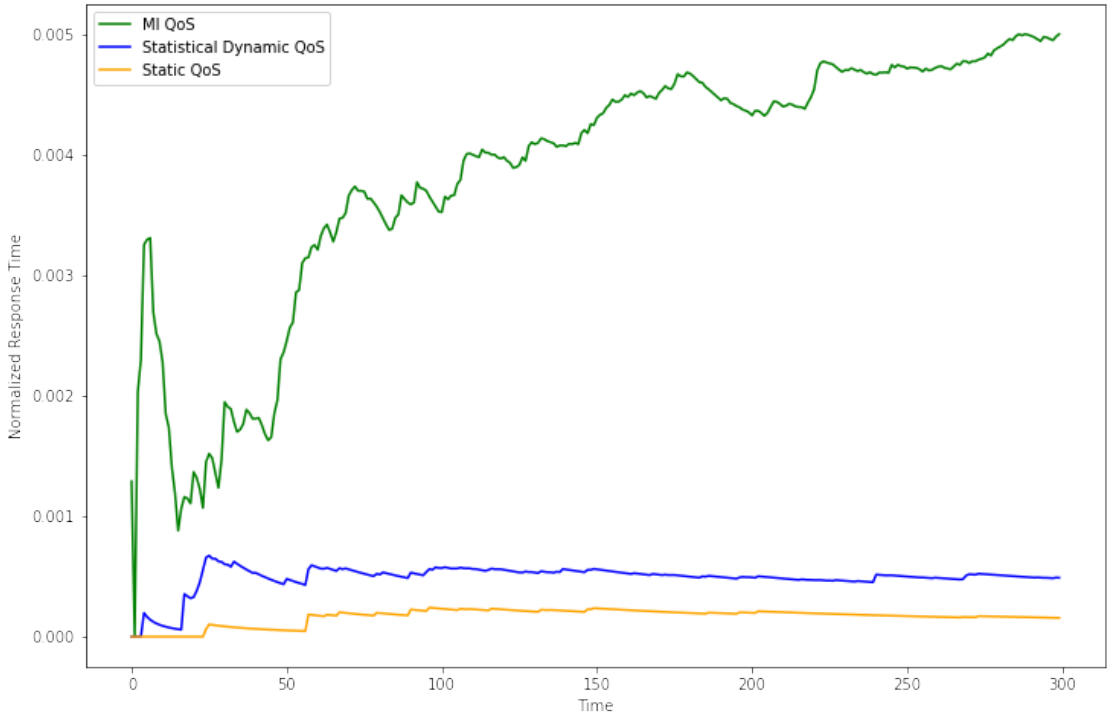


Figure 20: Response time Vs Time Using Static, Statical_dynamic and ML-PRegresser(Mean=0.05,Std=0.01)



Based on all the graphs of power consumption, reliability, and response time, we found that MLPregressor and Statical Dynamic QoS approaches show promising results in power consumption and reliability, and the Static QoS approach shows promising results in response time.

The LP-MQTT protocol's energy consumption is not static. It varies with the QoS level adjustments made by dynamic and machine learning-assisted strategies in response to changing network conditions. Through these strategies, LP-MQTT seeks to balance the adjustment between ensuring reliable message delivery and conserving energy, which is particularly critical for battery-powered IoT devices.

The trade-off between energy consumption and message delivery guarantee is critical in LP-MQTT. Lower QoS levels conserve energy but are less reliable, while higher QoS levels increase reliability at the cost of higher energy usage. Applications that need guaranteed delivery and can afford energy consumption will favor higher QoS levels. In contrast, applications with energy constraints, where message loss can be tolerated, may select for lower QoS levels. The LP-MQTT protocol offers a flexible and energy-efficient way of adjusting these QoS levels dynamically according to the current network conditions and application requirements.

6 CONCLUSION

While showing similarities in power consumption, reliability, and response time given against different probability of failure circumstances, detail perceived strengths and adjustment with each QoS strategy under LP-MQTT protocol. From the graphical depictions and findings from these graphs, it is possible to draw essential lessons that will be important for optimizing the performance of IoT systems in various network conditions.

Dynamic QoS can reduce power usage by adapting QoS levels according to real-time network conditions and application requirements, avoiding unnecessary power drain from high QoS levels when not needed. The system can respond to fluctuating network conditions, such as variable connectivity and bandwidth, by adjusting QoS levels to maintain message delivery guarantees without extreme power consumption. For battery-operated IoT devices, conserving power is critical, and dynamic QoS adjustment helps to extend their operational life by minimizing unnecessary energy expenditure. Dynamic QoS seeks an optimum balance between energy usage and message delivery reliability, which can be particularly beneficial in applications with varying criticality levels for different messages. In scenarios where network conditions decline, dynamic QoS can raise the QoS level to increase the likelihood of message delivery, which is crucial for critical IoT applications.

Implementing a system that can dynamically adjust QoS levels requires additional logic and possibly more complex programming, increasing the IoT system's complexity. Suppose the predictive model is not accurate or overly responsive to minor changes. In that case, it may fluctuate QoS levels unnecessarily, leading to increased message overhead and potentially more power consumption rather than less. Higher QoS levels, while more reliable, can improve communication latency due to the need for message acknowledgments and potential retransmissions.

Evaluating network conditions and making dynamic adjustments require processing capabilities and memory, which may be limited on some low-power IoT devices. Sensors and devices using dynamic QoS adjustment require continuous updates and maintenance to ensure the algorithms function correctly, incurring additional costs. The effectiveness of dynamic QoS adjustments is highly dependent on the accuracy of the predictive models being used, which rely on historical data that might not accurately predict future conditions. Frequent switching between QoS levels might lead to communications insta-

bility, potentially negatively impacting user experience and overall system performance. In summary, while dynamic QoS adjustment in LP-MQTT can significantly benefit energy efficiency and adaptability, it also introduces complexity.

Part III

APPENDIX

BIBLIOGRAPHY

[1] Marwa O Al Enany, Hany M Harb, and Gamal Attiya. A comparative analysis of mqtt and iot application protocols. In *2021 International Conference on Electronic Engineering (ICEEM)*, pages 1–6. IEEE, 2021.

[2] Anwar D Alhejaili and Omar H Alhazmi. Lightweight algorithm for mqtt protocol to enhance power consumption in healthcare environment. *Journal on Internet of Things*, 4(1):21, 2022.

[3] Mohsen Hallaj Asghar and Nasibeh Mohammadzadeh. Design and simulation of energy efficiency in node based on mqtt protocol in internet of things. In *2015 International conference on green computing and internet of things (ICGCIoT)*, pages 1413–1417. IEEE, 2015.

[4] Edgaras Baranauskas, Jevgenijus Toldinas, and Borisas Lozinskis. Evaluation of the impact on energy consumption of mqtt protocol over tls. In *CEUR workshop proceedings: IVUS 2019 international conference on information technologies: proceedings of the international conference on information technologies, Kaunas, Lithuania, April 25, 2019*, volume 2470, pages 56–60. CEUR-WS, 2019.

[5] Paolo Bellavista and Alessandro Zanni. Towards better scalability for iot-cloud interactions via combined exploitation of mqtt and coap. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6. IEEE, 2016.

[6] Fu Chen, Yujia Huo, Jianming Zhu, and Dan Fan. A review on the study on mqtt security challenge. In *2020 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 128–133. IEEE, 2020.

[7] Dan Dinculeană and Xiaochun Cheng. Vulnerabilities and limitations of mqtt protocol used between iot devices. *Applied Sciences*, 9(5):848, 2019.

[8] Heriberto J Jara Ochoa, Raul Peña, Yoel Ledo Mezquita, Enrique Gonzalez, and Sergio Camacho-Leon. Comparative analysis of power consumption between mqtt and http protocols in an iot platform designed and implemented for remote real-time monitoring of long-term cold chain transport operations. *Sensors*, 23 (10):4896, 2023.

[9] Arun Kumar, Sharad Sharma, Nitin Goyal, Aman Singh, Xiaochun Cheng, and Parminder Singh. Secure and energy-efficient smart building architecture with emerging technology iot. *Computer Communications*, 176:207–217, 2021.

[10] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, and Hongtaek Ju. Correlation analysis of mqtt loss and delay according to qos level. In *The International Conference on Information Networking 2013 (ICOIN)*, pages 714–717. IEEE, 2013.

[11] Jiaying Lin, Peng Wang, Jichuan Zhang, Zhiming Zhang, and Haoyang Sun. Plug and play technology for power distribution terminal management based on the iot ideas. In *2019 IEEE 8th International Conference on Advanced Power System Automation and Protection (APAP)*, pages 196–200. IEEE, 2019.

[12] Yifeng Liu and Eyhab Al-Masri. Evaluating the reliability of mqtt with comparative analysis. In *2021 IEEE 4th International Conference on Knowledge Innovation and Invention (ICKII)*, pages 24–29. IEEE, 2021.

[13] Edoardo Longo and Alessandro EC Redondi. Design and implementation of an advanced mqtt broker for distributed pub/sub scenarios. *Computer Networks*, 224:109601, 2023.

[14] F Luthfi, EA Juanda, and I Kustiawan. Optimization of data communication on air control device based on internet of things with application of http and mqtt protocols. In *IOP Conference Series: Materials Science and Engineering*, volume 384, page 012009. IOP Publishing, 2018.

[15] MV Masdani and Denny Darlis. A comprehensive study on mqtt as a low power protocol for internet of things application. In *IOP Conference Series: Materials Science and Engineering*, volume 434, page 012274. IOP Publishing, 2018.

[16] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE, 2017.

[17] Pegah Nikbakht Bideh, Jonathan Sönnerup, and Martin Hell. Energy consumption for securing lightweight iot protocols. 2020.

[18] Shital Pawar, Nibedita Panigrahi, AP Jyothi, Meghana Lokhande, Deepali Godse, and DB Jadhav. Evaluation of delay parameter of mqtt protocol.

[19] Ousmane Sadio, Ibrahima Ngom, and Claude Lishou. Lightweight security scheme for mqtt/mqtt-sn protocol. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 119–123. IEEE, 2019.

[20] Shubhangi A Shinde, Pooja A Nimkar, Shubhangi P Singh, Vrushali D Salpe, and Yogesh R Jadhav. Mqtt-message queuing telemetry transport protocol. *International Journal of Research*, 3 (3):240–244, 2016.

[21] Daniel Silva, Liliana I Carvalho, José Soares, and Rute C Sofia. A performance analysis of internet of things networking protocols: Evaluating mqtt, coap, opc ua. *Applied Sciences*, 11(11):4879, 2021.

[22] Martin Stusek, Krystof Zeman, Pavel Masek, Jindriska Sedova, and Jiri Hosek. Iot protocols for low-power massive iot: a communication perspective. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–7. IEEE, 2019.

[23] KALAIVANAN SUGUMAR. Mqtt-a lightweight communication protocol relative study. *Authorea Preprints*, 2020.

[24] Jevgenijus Toldinas, Borisas Lozinskis, Edgaras Baranauskas, and Algirdas Dobrovolskis. Mqtt quality of service versus energy consumption. In *2019 23rd International Conference Electronics*, pages 1–4. IEEE, 2019.

[25] Bharati Wukkadada, Kirti Wankhede, Ramith Nambiar, and Amala Nair. Comparison with http and mqtt in internet of things (iot). In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 249–253. IEEE, 2018.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

DECLARATION

Darmstadt, December 2022

Anchu Antony, Deepthi
Myladi Kelambath, February
2, 2024

.

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se