FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Ramisha Munir Awan

SEAMLESS CONNECTIVITY: INVESTIGATING IMPLEMENTATION CHALLENGES OF MULTIBROKER MQTT PLATFORM FOR SMART ENVIRONMENTAL MONITORING

Master's Thesis
Degree Programme in Computer Science and Engineering
June 2023

Awan R. (2023) Seamless Connectivity: Investigating Implementation Challenges of Multibroker MQTT Platform for Smart Environmental Monitoring. University of Oulu, Degree Programme in Computer Science and Engineering, 58 p.

ABSTRACT

This thesis explores the performance and efficiency of MQTT-based infrastructure Internet of Things (IoT) sensor networks for smart environment. The study focuses on the impact of network latency and broker switching in distributed multi-broker MQTT platforms. The research involves three case studies: a cloud-based multi-broker deployment, a Local Area Network (LAN)-based multi-broker deployment, and a multi-layer LAN network-based multi-broker deployment. The research is guided by three objectives: quantifying and analyzing the latency of multi-broker MQTT platforms; investigating the benefits of distributed brokers for edge users; and assessing the impact of switching latency at applications. This thesis ultimately seeks to answer three key questions related to network and switching latency, the merits of distributed brokers, and the influence of switching latency on the reliability of end-user applications.

TABLE OF CONTENTS

FOREWORD

I express my gratitude to my supervisors, Susanna Pirttikangas and Lauri Loven, for their unwavering support and the numerous hours they invested in supervising me. Their advice and suggestions were invaluable in refining all aspects of my final work, making the entire process smooth and enjoyable.

I am also deeply grateful to Timo Ojala, UBICOMP'S Interactive Edge research group, and the University of Oulu for providing me with a valuable study opportunity and an unforgettable life experience in Finland.

Furthermore, I extend my heartfelt thanks to my family and friends, whose meticulous care and warm companionship made my two-year journey living abroad truly special.

Oulu, June 16th, 2023

Ramisha Munir Awan

LIST OF ABBREVIATIONS AND SYMBOLS

MQTT	Message Queuing Telemetry Transport
Pub-sub	Publish and Subscribe
Cl	Client
NL	Newtrok Latency
SL	Switching Latency
IoT	Internet of things
LAN	Local area network
WAN	Wide area Network
HRLL	High-reliability, low-latency
WSNs	Wireless sensor Nertwork
QoS	Quality of Service
CBR	Cluster-based routing
VM	Virtual Machines
SN	Sensor Nodes
VR	Virtual Reality
AR	Augmented Reality
UA	User application
CN	Constraint network
PCB	Printed Circuit Board
AQ	Air Quality Detector Sensor
SD	Sound Detection Module
TH	Temperature and Humidity Sensor Module
KY	light sensor module

L_n	Network Latency
L_s	Switching Latency
L_a	Publisher network latency
L_b	Internet latency to the main server
L_c	Central server latency
L_d	Transmission latency back to the client
L_e	Client network latency
$L_{c'}$	Central server latency - Multi-layer
$L_{d'}$	Transmission latency back to the client- Multi-layer
$L_{e'}$	Client network latency- Multi-layer
TL	Total Latency
$ToMR$	Time of Message Receipt (Client)
$ToMS$	Time of Message Sending (Synced with Client's Time)

1. INTRODUCTION

1.1. Background

The Internet of Things (IoT) has experienced significant growth in various domains, including healthcare, smart cities, and transportation [1]. IoT relies on a vast network of interconnected devices, commonly referred to as sensors, which generate and transmit data. One of the primary challenges in IoT systems is the efficient and reliable distribution of data from sensors to subscribers. This is typically achieved using a broker-based communication model, wherein sensors publish data to brokers, which then distribute the data to the subscribers [2].

Mobile users play an essential role in the IoT ecosystem, consuming data from sensors and interacting with IoT devices. As these users move within the coverage areas of different brokers, it is crucial to ensure that they maintain connectivity to the nearest broker to minimize latency and message loss. This process of disconnecting from one broker and connecting to another is known as broker switching. However, during the broker switching process, there may be a delay in receiving data, which can significantly affect the overall performance and user experience.

Previous studies have investigated various aspects of IoT communication, such as data dissemination, network topology, and quality of service (QoS) [3]. However, there is limited research focusing on the specific problem of broker switching and its impact on latency and message loss in dynamic IoT environments. Therefore, this thesis aims to evaluate the latency during the broker switching process and propose potential improvements and solutions.

While broker-based communication models have been extensively studied, the rapid evolution of IoT technology and the increasing number of mobile users necessitate continuous research and development to address new challenges. Many studies have explored edge and fog computing to improve IoT systems' scalability, efficiency, and reliability [4]. Edge computing brings data processing closer to the source, minimizing latency and improving the overall user experience [5]. Fog computing extends the concept of edge computing by incorporating hierarchical processing and storage capabilities [6].

In 2019, a survey described edge computing and multi-access edge computing (MEC) to reduce latency and improve the user experience [7]. Similarly, another study investigated the use of machine learning algorithms to predict mobile user movements [8]. To address the aforementioned research gaps, there is a need for studying seamless handover between brokers and dynamic broker allocation based on the user's location and mobility patterns and proactively initiating broker switching processes, minimizing latency and ensuring continuous connectivity.

These recent studies highlight the ongoing efforts to address the challenges associated with broker switching in dynamic IoT environments. However, further

research is necessary to develop more efficient, secure, and scalable solutions that can adapt to the ever-evolving IoT landscape.

1.2. Problem Statement

The current state of the art in smart environmental monitoring relies on a multi-broker MQTT platform. However, this platform has several implementation challenges, including high switching latency and scalability. This research will investigate these challenges and analyze their impact on the performance and scalability of multi-broker MQTT platforms for smart environmental monitoring.

1.2.1. Hypothesis:

The main hypothesis of this thesis is that multi-MQTT broker platforms will have similar switching delays behavior as with traditional network delays at both the cloud and edge levels. After validation of this hypothesis, the performance implications of distributed multi-edge MQTT brokers and the potential considerations for deploying end-user applications will be evaluated.

1.2.2. Objectives and Research Questions:

The primary objectives of this research are:

1. Measurement and Analysis of Latency (Network and Switching): The first objective of this study is to measure and analyze the latency of multi-broker MQTT platforms. This objective will be achieved through the following tasks:
    - Conducting experiments to quantify the latency in different scenarios.
    - Comparing and evaluating the latency performance of various multi-broker MQTT platforms. The findings of this objective will provide insights into the actual latency experienced in multi-broker MQTT platforms.
2. Assessment of Hypothesis in terms of Latency and System Performance:

    The second objective focuses on assessing the impact of switching latency on the performance of multi-broker MQTT platforms. The following tasks will be undertaken to achieve this objective:

    - Evaluating the performance metrics, using the message travel time of multi-broker MQTT platforms under device performance.
    - Investigating the relationship between switching latency and the overall performance and responsiveness of the system. Through this objective, the research will provide an understanding of how switching latency influences the performance of multi-broker MQTT platforms and the feasibility of end-user application.

From these objectives, the following research questions will be explored.

(a) What is the network and switching latency of multi-broker MQTT platforms?

(b) How does switching latency impact the reliability of end-user applications on multi-broker MQTT platforms?

By accomplishing these objectives, this research study aims to contribute to the knowledge and understanding of switching latency in multi-broker MQTT platforms, explore the benefits of distributed brokers for end users, and evaluate the impact of switching latency on the overall performance of multi-broker MQTT platforms.

1.3. Scope and Limitations:

This research aims to thoroughly examine the challenges related to switching latency and scalability in multi-broker MQTT platforms employed for smart environmental monitoring. The scope of the study is confined to the following aspects:

- **Latency Analysis:** The study will focus primarily on the latency aspects of multi-broker MQTT platforms. This includes both network and switching latency. Various multi-broker MQTT platforms will be explored and their latencies will be measured, compared, and evaluated under different scenarios.
- **Distributed Brokers:** The benefits offered by distributed brokers for the end users will be carried out within the context of multi-broker MQTT platforms. The impact of these distributed brokers on the reliability, responsiveness, and scalability of the platform will also be explored. However, the study will not venture into the specific implementation details of these distributed systems.
- **Switching Latency Impact:** The research will investigate the impact of switching latency on the overall system performance and reliability. The relation between switching latency and overall system responsiveness will be studied, along with the implications for end-user applications. Specific device performance scenarios will be taken into account for these investigations.
- **End-User Applications:** Although the study will evaluate the potential implications of the distributed multi-edge MQTT brokers on end-user applications, the development or implementation of these applications is not within the scope of the research.

The limitations of this study include: While this study aims to provide valuable insights into the challenges and performance aspects of multi-broker MQTT platforms in smart environmental monitoring, it is essential to acknowledge the following limitations:

1. The study will not evaluate the impact of other factors that could affect system performance, such as hardware limitations.

2. The study will not address issues related to security or privacy in IoT systems.
3. The study will be limited by the available resources and time frame for conducting the research.

   Despite these limitations, this study provides valuable insights into switching latency, benefits of distributed brokers, and the impact of switching latency on performance on edge applications deployed on multi-broker MQTT platforms.

1.4. Organization of the Thesis

Chapter 2 provides an extensive review of the relevant literature, discussing the current state of IoT and sensor networks, the role of brokers in communication systems, handover mechanisms in mobile networks, and previous research addressing broker switching.

Chapter 3 details the system model and methodology used in the research. It includes three case studies: a cloud-based multi-broker deployment, a LAN-based multi-broker deployment, and a multi-layer LAN network-based multi-broker deployment. The methodology involves implementing the system model, collecting data, analyzing the data, and using an optimized approach to calculate latency. The assumptions and constraints section discusses the factors that may affect the system's behavior. Metrics for evaluation, such as are defined, and the research methodology outlines the approach taken to address the research questions.

Chapter 4 describes the simulation and experimental setup used to evaluate the latency during broker switching. This includes the simulation environment, broker and sensor configuration, and test scenarios.

Chapter 5 presents the results and analysis of the research. The latency analysis examines the time taken for a user to disconnect from one broker and connect to a new broker. Factors affecting performance are discussed, and the findings are compared with existing approaches in the literature.

Chapter 6 concludes the thesis by summarizing the main findings and discussing the implications of the research. The latency broker switching are evaluated, and potential improvements and solutions are proposed. Possible future research directions include investigating more advanced broker selection algorithms, exploring other handover techniques, and conducting experiments with larger-scale IoT systems and different scenarios.

2. LITERATURE REVIEW

2.1. Internet of Things (IoT) and Sensor Networks

The Internet of Things (IoT) represents a paradigm shift in technology, enabling countless devices and sensors to communicate and exchange data over the internet [9]. The primary aim of IoT is to streamline data sharing, analysis, and automation across various applications such as environmental monitoring, healthcare, smart homes, security, and industrial automation [10].

The Significance of Sensor Networks in IoT: Sensor networks play an important role within the Internet of Things (IoT) context, consisting of spatially distributed sensors cooperatively monitoring physical or environmental conditions [11]. The data they collect is transmitted via IoT platforms, commonly employing a broker-based communication model [12].

MQTT Protocol in Dynamic IoT Environments: The MQTT (Message Queuing Telemetry Transport) protocol is one of the most prevalent protocols in this model due to its efficiency and reliability [13]. In dynamic IoT environments with mobile users, maintaining seamless connectivity to the nearest broker is crucial, typically achieved through broker switching. However, broker switching can introduce latency and message loss, adversely affecting the user experience and overall system efficiency. Therefore, the impact of network latency and the switching process when changing brokers in dynamic IoT environments requires thorough investigation and research.

Impact of Network Latency and Broker Switching: The integration of mobile users into IoT environments adds complexity, introducing variables such as fluctuating connection quality and changes in proximity to different brokers [14]. Thus, effective handover mechanisms in mobile networks [15] are essential to maintain seamless connectivity and optimal system performance. Despite the potential benefits of IoT and sensor networks across various fields, they face implementation challenges, particularly in maintaining seamless connectivity in dynamic environments [16]. This research aims to address these challenges by focusing on broker switching and its impact on latency and message loss. It is worth noting that other factors, such as network congestion and hardware limitations, could also impact system performance [16]. Additionally, this research does not address security or privacy issues in IoT systems, which are significant areas of concern [17].

2.1.1. Challenges of Achieving HRLL in Multibroker MQTT Platforms

This section will discuss the performance constraints and high-reliability, low-latency (HRLL) requirements of wireless IoT networks. In the domain of IoT communication, an exhaustive review conducted by the authors [18] focused on wireless IoT network communication and performance constraints related to reliability and latency. Their comprehensive analysis explored strategies to achieve high-reliability and low-latency (HRLL) in wireless IoT networks across the physical, MAC, and network layers. By conducting a meticulous review of over 150 articles, the authors aimed to uncover the fundamental performance limitations and potential strategies to address these limitations, thereby achieving HRLL in wireless IoT networks. The review examined

various techniques employed in practical scenarios such as industrial automation, V2X communication, and smart grids. It also delved into the standardization efforts for HRLL IoT communication. Additionally, the authors evaluated information-theoretic limits for HRLL communications at the physical layer, and optimization strategies such as power resource management and spectrum optimization for the MAC layer. At the network layer, they considered optimized network structure, efficient traffic allocation, and coding schemes to reduce latency and improve reliability. The authors concluded that an ideal scenario for HRLL IoT communication should include factors such as low latency, high reliability, increased throughput, fading robust capability, and support for mobilization. This holistic and in-depth review offers valuable insights for our study, providing a robust understanding of the performance limitations and potential solutions, which will be integral to our investigation on broker-switching in multi-broker MQTT platforms within dynamic IoT environments.

2.1.2. IoT and Its Potential Impact

A collaborative paper by Silverio-Fernandez et al. [19] offers an expansive overview of IoT and its potential impact on various life domains. The authors portray the IoT as a complex network of interconnected "things," including machines, objects, environments, and infrastructures. They emphasize the role of embedded processing, which provides intelligence to these smart machines, enabling them to generate vast amounts of data. This data is then processed to yield valuable insights, enable useful actions, and enhance control over various aspects of our lives. The overarching objective is to facilitate an easier, safer, and more environmentally friendly way of living. This comprehensive perspective on IoT is highly relevant to our study as we delve into exploring the impact of IoT on environmental monitoring using sensor networks and MQTT platforms.

Clear Definition of Smart Devices in IoT: The authors of paper [20] address the issue of inconsistent terminology frequently encountered in literature when describing devices in the IoT domain. They propose the term "smart device" as a starting point to establish a clear definition for devices in IoT. Through a systematic review of the existing literature, they explore the concept and key features of smart devices, highlighting their role in IoT. The authors identify smart devices as the primary interconnected objects in IoT, playing a pivotal role in the operation of this paradigm. The definition of a smart device is defined by three key features: context-awareness, autonomy, and device connectivity. Additionally, they mention other features like mobility and user interaction, which were less prominent in the literature and thus not considered essential due to the nature of IoT as a network primarily focused on device-to-device connectivity. This understanding and categorization of smart devices is important for our research, which relies on smart sensors and their communication within an IoT paradigm.

Implications and Challenges of IoT Applications Rahmani et al. [21] present a refined view of the IoT, focusing on its implications and challenges. They discuss the potential benefits, open issues, threats, and limitations associated with IoT applications. The authors place a strong emphasis on factors such as cost, real-time responsiveness, and security when evaluating and developing IoT applications.

Through a comprehensive assessment of existing literature, which involved reviewing numerous articles and addressing six analytical questions, they found that the most influential factor for evaluating IoT applications is cost, mentioned in 79% of the reviewed articles. Real-time responsiveness was emphasized in 64% of the articles, while security and error issues were discussed in 57% of the reviewed articles. Their approach to analyzing and categorizing the existing literature offers a useful framework for evaluating the design and implementation of our proposed multi-broker MQTT platform, particularly in terms of cost, responsiveness, and error handling.

Security and Privacy Challenges in IoT Security and privacy challenges in the IoT domain are addressed in-depth by the researchers [22]. The authors emphasize that despite the growing prevalence of IoT and its profound impact on various aspects of human life, safeguarding user security and privacy continues to pose a significant research challenge. They argue that conventional security solutions developed for centralized and distributed systems are not directly suitable for IoT systems due to their distinctive characteristics, such as intermittent connectivity, extensive scalability, dynamic changes, and limited resources. In their study, the researchers provide an in-depth overview of security and privacy within the context of IoT systems, identifying four key characteristics that introduce unprecedented threats and challenges to existing security solutions. These characteristics encompass scalability, connectivity, limited resources, and dynamic changes. This exploration of security and privacy issues is important for our study as it raises important considerations for the design and implementation of our multi-broker MQTT platform in a dynamic IoT environment.

2.1.3. Intelligent Routing in IoT-Based Wireless Sensor Networks

The research work presented in paper [23] focuses on Wireless Sensor Networks (WSNs) in the context of IoT. WSNs are used for sensing environmental variables, collecting data, and transmitting it to a base station for analysis. In IoT-based WSNs, intelligent routing plays a crucial role in enhancing the Quality of Service (QoS) in the network. The researchers propose a Neuro-Fuzzy Rule-Based Cluster Formation and Routing Protocol specifically designed for efficient routing in IoT-based WSNs. Experimental results demonstrate that proposed routing algorithm outperforms existing approaches in terms of energy utilization, packet delivery ratio, delay, and network lifetime. The findings from this research offer potential techniques for improving the efficiency of sensor data routing in our study, which could help minimize latency and message loss during broker switching.

The study presented in paper [24] introduces a novel three-factor authentication scheme that generates session keys for WSNs. The security properties of the proposed scheme are formally verified using Proverif, demonstrating its ability to maintain security requirements. the informal analysis indicates that proposed scheme is practical and fulfills general security needs. the proposed scheme excels in terms of security and is well-suited for real-world applications. the performance is evaluated by conducting simulations using NS-3 in an IoT environment. This focus on security in WSNs is highly relevant to our research as it raises important considerations for maintaining secure communications when implementing our multi-broker MQTT platform in a dynamic IoT environment.

CBR-ICWSN: Cluster-Based Routing Protocol for ICWSN The paper by authors [25] introduces a cluster-based routing (CBR) protocol called CBR-ICWSN for information-centric wireless sensor networks (ICWSN) in IoT. The proposed protocol utilizes a black widow optimization (BWO) based clustering technique to effectively select optimal cluster heads (CHs). By leveraging Information-Centric Networking (ICN) architecture and incorporating optimization techniques, the CBR-ICWSN protocol enhances data access mechanisms, reliability during mobility events, and reduces maximum delay in multihop communication. This paper provides useful insights on optimization techniques that could be used in our research to reduce latency and message loss during broker switching.

Efficient Routing for Wireless Sensor Network-Based IoT Applications The research presented in paper [26] addresses the challenges faced by wireless sensor network-based IoT applications, including end-to-end delay, packet loss, and reduced node lifetime due to energy depletion. An efficient routing protocol is proposed to tackle these challenges and enhance the Quality of Service (QoS) in the network. The proposed routing protocol takes into account three factors: lifetime, reliability, and traffic intensity at the next-hop node. These considerations are highly relevant in our study as we explore the effects of broker switching on latency and network latency. Moreover,[27] focuses on data dissemination in WSNs within an IoT system. The proposed Optimized Energy Management Model for Data Dissemination (OEM-DD) framework offers a novel approach to optimizing energy consumed during data transmission among all nodes in an IoT system's sensor network. This approach to energy optimization is highly relevant to our study, which seeks to maintain efficient and reliable communication in a dynamic IoT environment with mobile sensors, especially during MQTT broker-switching events.

Lightweight Synchronization Algorithm for WSNs Paper [28] presents a lightweight synchronization algorithm specifically designed for WSNs, demonstrating its effectiveness in synchronizing data measurements on a per-hop basis without causing significant overhead. Instead of synchronizing the clocks of the nodes themselves, the algorithm focuses on the synchronization of measurements as they are transmitted in data packets towards the sink node.

The proposed algorithm takes into account factors such as residual time (the time period a data packet remains within a node), distance (number of hops to the sink node), and average skew deviation. This methodology provides an innovative perspective on ensuring synchronized data transmission within a WSN, thereby contributing to the overall efficiency and reliability of the IoT system. In the context of dynamic IoT environments where broker-switching events may introduce additional complexity, such a lightweight synchronization approach may prove beneficial. The focus on energy efficiency and robustness is particularly applicable to our study, as these are key considerations in the optimization of MQTT broker-switching processes. The in-depth exploration of IoT and sensor networks in this section provides a strong foundation for our investigation into MQTT and broker-switching challenges in multi-broker systems. It equips us with a comprehensive understanding of the existing systems, their performance limitations, and potential solutions, enabling us to more effectively address the specific issue of broker-switching latency and network latency in dynamic IoT environments. Additionally, it provides a broader context within which

we can position our research findings and solutions, emphasizing their significance and potential impact in the realm of IoT communication.

To summarize, the collective body of work on IoT and sensor networks presents a comprehensive overview of the existing methodologies, applications, challenges, and future directions in the domain. It highlights the complexity of IoT ecosystems, the integral role of smart devices within these systems, the significant factors influencing the evaluation of IoT applications, and the crucial importance of security and privacy in the IoT realm. Additionally, it underlines the significance of efficient routing and secure communications in IoT-based WSNs, emphasizing the need for innovative protocols and strategies to enhance data access mechanisms, reliability, and energy efficiency in the face of mobility events and dynamic environmental conditions. These insights serve as the foundation for the subsequent sections of this study, which delve into the refinement of MQTT, a publish-subscribe protocol central to IoT communication, and broker-switching in multi-broker MQTT platforms within dynamic IoT environments. Understanding the performance limitations and potential solutions across the three core network layers (physical, MAC, and network layers), as well as the unique characteristics and challenges posed by IoT systems and IoT-based WSNs, will enable us to develop a comprehensive and effective framework for broker-switching that addresses the key issues of latency, message loss, and overall system reliability. Moreover, the broad overview of IoT presented in this section will allow us to contextualize our research findings and solutions within the wider IoT landscape, and effectively communicate the significance and potential impact of our work.

2.2. Publish-Subscribe Systems

This section reviews relevant literature that explores various aspects of publish-subscribe systems and their applicability to the research topic of investigating implementation challenges of a multibroker MQTT platform for smart environmental monitoring.

A critical survey and taxonomy of publish-subscribe systems are provided in [29]. This study focuses on various design features and technologies employed in publish-subscribe systems, including message queues, clusters, databases, message delivery policies, federations of brokers, security services, communication protocols, and streaming APIs. The concepts of message queuing and publish-subscribe systems are discussed along with publish-subscribe protocols for cloud and IoT. The evaluation focuses on seven state-of-the-art open-source systems: Apache Kafka, Orion-LD, RabbitMQ, Pushpin, Stellio, Scorpio, and Faye. Understanding the design features and technologies of publish-subscribe systems can help in identifying the most suitable components and protocols to minimize latency.

In the study presented in [30], a security framework for topic-based publish/subscribe systems in the IoT is introduced. Although the primary focus is on security and privacy at the broker level, the framework's loose coupling and attribute-based encryption (ABE) can be leveraged to ensure secure and authorized data transmission during the broker switching process. By incorporating ABE, fine-grained access control can be achieved based on users' attributes and data, allowing only authorized users with specific attributes to access the data. This attribute-

based approach can be utilized to enhance the security and privacy aspects of broker switching in the multibroker MQTT platform.

Another study [31] proposes a hierarchical architecture for publish/subscribe systems, consisting of a back-end big data cluster and a distributed broker network. This architecture can be adapted to manage user subscriptions and facilitate the delivery process in a multibroker MQTT platform. The optimization problem presented in the paper aims to minimize the maximum load on brokers, which is vital for efficient broker switching. By leveraging the concepts of initial placement of brokers, dynamic subscriber migration, and occasional shuffles, the proposed optimization approach can help provide basis for building MQTT platform.

In [32], the challenges of authentication and confidentiality in content-based publish/subscribe systems are addressed. Although the study primarily focuses on content-based routing, the proposed approach for ensuring authentication of publishers and subscribers and maintaining event confidentiality can be adapted to enhance the security and privacy aspects of the broker switching process.

Dynamic entity summarization in publish/subscribe systems is discussed in [33], which presents an abstractive publish/subscribe summarization system for dynamic IoT environments. While abstraction may introduce some latency, it significantly reduces the number of notifications. Incorporating the proposed abstractive summarization system can help minimize the message volume.

The study described in [34] introduces a peer-to-peer type of topic-based publish/subscribe model. Although the focus is on decentralized information exchange, the prevention of unauthorized information flow and illegal object delivery can be relevant to the security aspects. The TOBS ((topics-of-objects–based synchronization)) protocol proposed in the paper can ensure that only authorized objects are delivered to target peers, aligning with the objective of maintaining secure and reliable connections.

Information-Centric Networking (ICN) is explored in the context of IoT in [35]. The paper discusses the advantages of request-driven nature and unsolicited link-local signaling in restricting attack surface while enhancing responsiveness. The HoP-and-Pull (HoPP) publish-subscribe scheme, designed for IoT scenarios with low-power and lossy wireless networks, can provide insights into efficient and reliable data transmission. The evaluation of HoPP in different scenarios can inform the development of efficient multibroker MQTT platform.

Structured peer-to-peer networks, specifically distributed hash tables (DHTs), are investigated for decentralized and scalable publish/subscribe systems in [36]. The authors proposed OpenPubSub is an innovative peer-to-peer content-based publish/subscribe system that operates on an approximate semantic framework. It employs a hybrid event routing model that merges rendezvous routing and gossiping within a structured peer-to-peer network. Distinct from conventional logical key spaces, the network is constructed using a high-dimensional semantic vector space. The experimental results demonstrate the effectiveness of proposed approach. By employing rendezvous-based routing, the system achieves recall rates of up to 54% and simultaneously reduces messaging overhead by 44%. Furthermore, the hybrid routing approach maintains recall rates of up to 43.8%, with a messaging overhead reduction of 59%. These results can be valuable in optimizing the efficiency and reliability in multibroker MQTT platform.

In [37], the limitations of centralized architectures in publish-subscribe messaging models are addressed by proposing Trinity, a distributed publish-subscribe broker with Byzantine fault tolerance and blockchain-based immutability. Trinity ensures fault tolerance and data integrity by distributing data published by publishers to all brokers using blockchain technology. Leveraging consensus protocols and distributed ledger technology, Trinity enhances the security and reliability aspects of the publish-subscribe messaging model. The concepts presented in this study can be applied to ensure secure and reliable data transmission.

The PoSSUM system, introduced in [38], aims to provide user-friendly entity summaries in publish/subscribe systems. By incorporating data integration, diverse entity summarization, reasoning rules, and a Triple2Rank scoring mechanism, PoSSUM achieves significant message reduction. The evaluation results demonstrate the effectiveness of PoSSUM in providing high-quality entity summaries. Adopting the entity-centric approach and diverse summarization techniques can be helpful in building multibroker MQTT platform.

Overall, the reviewed literature on publish-subscribe systems provides insights into various aspects of secure data transmission, latency reduction, message loss mitigation, and decentralized architectures. By incorporating relevant concepts and techniques from these studies, the implementation challenges of a multibroker MQTT platform for smart environmental monitoring can be better addressed, enhancing the seamless connectivity and overall system performance.

2.3. Advances in IoT System Optimization

BMDD: Enhancing IoT Performance through Broker-less and Microservice Architecture:A notable study [39] introduced BMDD (Broker-less and Micro service architecture, Decentralized identity, and Dynamic transmission messages), an IoT Platform specifically designed to tackle the challenges faced in IoT systems. BMDD integrates broker-less and micro service architectural models, decentralized identity, and dynamic message transmission to optimize system performance. The platform addresses issues such as single failure points by decentralizing authentication and implementing Role-Based Access Control (RBAC). Furthermore, BMDD leverages the gRPC protocol and Kafka to enhance transmission rates while reducing power consumption compared to traditional MQTT protocols. The development of a dynamic message transmission mechanism enables seamless communication among devices.

Secure Device-to-Device Communications: The rapid deployment of IoT has led to need for secure device-to-device (D2D) communications [40]. In this paper, the authors propose secure versions of MQTT and MQTT-SN protocols, namely SMQTT and SMQTT-SN, respectively, to address the security concern. The proposed protocols enhance existing MQTT protocol by incorporating security features based on Key/Ciphertext Policy-Attribute Based Encryption (KP/CP-ABE) using lightweight Elliptic Curve Cryptography. To validate the feasibility of SMQTT and SMQTT-SN for various IoT requirements, authors conduct simulations and evaluate their performance. The simulations provide insights into security and efficiency of the protocols in different IoT scenarios.

Edge-based MQTT Broker Cluster: This paper [41] focuses on implementation of an edge-based MQTT broker cluster using lightweight container orchestration and single board computers. The objective is to create a cost-effective, scalable, and lightweight messaging solution for IoT devices in remote areas with limited computational resources and network connectivity. To support distributed MQTT brokers, an intermediate layer of cluster servers is developed to facilitate collective processing. The paper also presents evaluations of the MQTT broker cluster, including message throughput, end-to-end latency, and runtime performance. The results show that broker cluster maintains an average latency of less than 10 milliseconds and a worst-case latency of 52 milliseconds.

Brokerless Pub/Sub Architecture: This paper [42] focuses on a real-time water quality monitoring system that utilizes a brokerless publisher-subscriber (pub/sub) architecture framework. The system employs sensors to measure various water metrics such as temperature, pH, and dissolved oxygen level. The collected data is stored in a database and processed stochastically for further analysis of water quality. The authors performed an experiment that compares proposed pub/sub architecture with MQTT. The experiment demonstrates superior performance of proposed architecture in terms of network latency and throughput, particularly for diverse message payload sizes.

SDN-Based Publish/Subscribe Brokers: This paper [43] focuses on efficient dissemination of events in smart city applications using publish/subscribe brokers. These brokers handle data from devices and distribute it to subscribers through named logical channels called topics. The use of SDN allow load balancing between brokers by switching topics. The paper proposed an algorithm for SDN controller that minimizes load difference between brokers while adhering to a reconfiguration limit, specifically for data and decision fusion applications. The problem of minimizing load difference within a reconfiguration budget, with the constraint of indivisible topics, is formulated as an Integer Linear Programming (ILP) problem. The proposed heuristic is evaluated using realistic simulation traffic traces and compared against a threshold-based baseline heuristic that relies on instantaneous statistics of topics. The results show that proposed heuristic achieves significantly better load distribution, up to 2000% improvement compared to baseline heuristic and reduces topic switching by at least 27%.

2.4. Fog-Based IoT

The researchers [44] addresses challenge of delivering real-time and predictable latency for IoT applications by integrating Cloud and Fog computing paradigms. The goal is to minimize overall service request latency by scheduling IoT service requests effectively. The researchers introduced a customized adaptation of the genetic algorithm (GA) as an optimized approach to effectively schedule IoT service requests. By considering the dynamic nature of the IoT environment, their simulation-based evaluation demonstrated significant improvements in overall latency, ranging from 21.9% to 46.6%, compared to other scheduling techniques. Additionally, the proposed approach shows a substantial enhancement in meeting request deadlines, with improvements of up to 31%.

Furthermore, Cui et al. [45] have examined the trade-off between latency and energy consumption in mobile edge computing (MEC), which is relevant to broker implementation challenges. They proposed a solution using the non-dominated sorting genetic algorithm (NSGA-II) to optimize these objectives. Their experiments, conducted on a simulator within the MATLAB platform, showcased the effectiveness of their approach in achieving better latency-energy trade-offs and improving system performance.

By considering the relevant research on fog-based IoT, latency estimation, and energy consumption, this study aims to contribute to the existing literature by providing insights into the specific challenges of broker implementation.

2.5. Optimizing Latency in Healthcare IoT: Leveraging MQTT Protocol

In the context of seamless connectivity for smart environmental monitoring, the concept of publish-subscribe systems and knowing where they are critical in terms of end applications (such as security and healthcare fields) is highly relevant. In healthcare IoT applications, where real-time data is crucial, minimizing different types of latency, including computation, network, and communication latency, becomes essential. To address this, fog computing has been proposed as a solution to reduce high latency. One approach [46] incorporates an analytical model and a hybrid reinforcement machine learning algorithm, specifically support vector machines, within a fog computing environment. This solution combines a fuzzy inference system with reinforcement learning and neural network evaluation strategies for data packet allocation and selection. The effectiveness of this approach was demonstrated using iFogSim and Spyder simulators, resulting in a latency reduction of 97% compared to state-of-the-art techniques.

Additionally, Alshammari et al. [47] propose a real-time remote patient monitoring system based on IoT to meet the increasing need for high-quality healthcare in overpopulated nations. The system leverages technological advancements to monitor vital signs remotely, ensuring accuracy and reliability. In order to reduce latency in transmitting real-time signals, the study adopts the MQTT protocol to transmit vital signals from the proposed method to a website. By incorporating MQTT, the system aims to optimize latency and facilitate timely data transmission for efficient remote patient monitoring in healthcare IoT applications.

These healthcare-focused examples illustrate the importance of addressing latency issues in IoT systems, including the use of protocols like MQTT to enable efficient data transmission. By investigating the specific challenges of broker switching and its impact on latency in IoT environments, this research aims to contribute to the existing body of knowledge on IoT system performance and inform future research in this field.

2.6. Performance Analysis of IoT Systems

2.6.1. Latency Estimation in IoT

In the context of our research on seamless connectivity and broker switching in IoT environments, it is essential to understand the estimation and management of latency. Latency refers to the time delay experienced in transmitting data between nodes in a network, and it plays a crucial role in maintaining optimal system performance and user experience. In this section, we review relevant literature on latency estimation in IoT and explore approaches and techniques that can be applied to our specific problem.

One study by the author [48] focused on network delay components and developed a framework for latency estimation in fog-based IoT using a network coordinate system (NCS). This framework accurately predicts the delay between end-to-end nodes and can be highly relevant to our research. The author conducted a case study involving 20 routers in different houses, observing an average round-trip time (RTT) of 500ms. Each router in the house was considered a fog node, and the mean latency between nodes was measured as 10ms, providing inter-fog latency data for experimentation. The author implemented the Fog-based IoT latency estimation (FILE) algorithm, achieving 90% accuracy with 13 or more landmarks. In comparison, the Global Network Positioning (GNP) algorithm required 16 landmarks for similar accuracy. These findings highlight the potential of latency estimation techniques in fog-based IoT scenarios and their relevance to our investigation of broker switching and its impact on latency.

In another survey by the authors [49], the focus was on discussing approaches to reduce latency in transmitting time-sensitive data in real-time for both cloud and IoT devices. While this survey did not specifically address broker switching or dynamic IoT environments, it provided valuable insights into techniques and technologies that can be employed to reduce latencies in IoT and cloud applications. The systematic review classified 23 approaches and 32 technologies associated with latency in the cloud and IoT domains, shedding light on the current state of research in latency reduction. By examining 112 papers on latency reduction, the survey identified existing research gaps and works in latency reduction for IoT, further emphasizing the importance of addressing latency issues in our research.

Addressing the high latency experienced by mobile devices (MDs) due to limited computing power and reliance on remote clouds, a paper by the authors [50] proposed the deployment of cloudlets (miniature clouds) at the network edge as a solution for reducing service latency. This approach aligns with our investigation and its impact on latency, as it aims to optimize latency performance in an IoT network. The paper presented a novel binary-based differential evolution cuckoo search (BDECS) algorithm for the selection of optimal permanent cloudlet deployment locations from a multitude of access points (APs) in the network. Extensive simulations demonstrated the superiority of the proposed algorithm in terms of minimizing cost and latency, showcasing its potential relevance to our research objectives.

Furthermore, the design and implementation of a service placement architecture for IoT, with a focus on the Service Orchestrator module, was discussed by Velasquez et al. [51]. This architecture addresses the requirement for low latency in new services and applications enabled by IoT and Smart Cities paradigms. The proposed smart service placement system efficiently locates services in optimal positions based on specific

needs. While the study's primary emphasis is not on broker switching or dynamic IoT environments, the concept of optimizing service placement to achieve low latency aligns with our research goals. The insights and implementation details provided in this paper can contribute to our understanding of latency-related challenges and potential solutions in the context of IoT and Smart Cities.

By exploring the literature on latency estimation in IoT and related domains, we gain valuable insights into techniques, algorithms, and frameworks that can be adapted and applied to our research on seamless connectivity, broker switching, and their impact on latency in dynamic IoT environments. The studies reviewed highlight the significance of addressing latency issues and provide a foundation for proposing potential improvements and solutions in our investigation.

2.6.2. Energy Consumption in IoT

Energy consumption is a critical aspect to consider in the context of seamless connectivity and the implementation of a multibroker MQTT platform for smart environmental monitoring. In dynamic IoT environments with mobile users, where broker switching is essential, energy efficiency plays a significant role in maintaining optimal system performance. This section explores relevant research on energy consumption in IoT and its implications for the proposed research topic.

One area of focus is the Clout of Things (CoT), which combines cloud computing with IoT to provide ease of management, ubiquitous access, service creation, discovery, and resource provisioning [52]. CoT offers potential benefits for energy management in IoT systems, as efficient resource management frameworks can be developed for IoT devices in the Fog. These frameworks can address resource prediction, customer-based estimation and reservation, advance reservation, and pricing based on customer characteristics. By optimizing resource utilization, energy consumption can be effectively managed, contributing to more sustainable and efficient energy usage in the IoT ecosystem.

Moreover, IoT technology has been applied to energy management systems to monitor and optimize energy consumption [53]. These systems typically incorporate microcontrollers, sensors, communication protocols, and cloud-based systems. By leveraging IoT technology, energy management practices can be enhanced, leading to more efficient and sustainable energy usage. Within the context of seamless connectivity, it is crucial to evaluate the energy consumption associated with broker switching and determine methods to minimize it without compromising system performance.

Energy consumption also comes into play when considering machine learning (ML) algorithms for tasks such as intrusion detection in smart home systems [54]. Traditional cloud-based ML approaches may introduce privacy and latency concerns, which can impact energy efficiency. Evaluating different ML approaches, including cloud computing-based ML, edge computing-based ML, and IoT device-based ML, becomes relevant for optimizing energy consumption during training and inference phases. By selecting energy-efficient algorithms and deploying them on appropriate devices, the overall energy consumption of the IoT system can be reduced.

To address the challenges of IoT sustainability and energy consumption, integrating edge computing into IoT platforms has been proposed [55]. Edge computing moves some of the computation closer to the edge devices, potentially reducing energy consumption. By implementing edge computing paradigms, IoT devices can benefit from localized processing and efficient resource utilization, leading to improved energy efficiency in the system.

Furthermore, in the healthcare domain, addressing energy consumption issues in wireless body sensor networks (WBSNs) is crucial for maintaining the longevity of battery-powered devices [56]. The use of cognitive cooperative communication schemes and cognitive master nodes can help mitigate energy wastage caused by collisions and retransmission processes. By optimizing network architecture, reducing collisions, and minimizing retransmission, energy consumption in WBSNs can be significantly reduced, prolonging the battery life of the devices.

Considering the insights from studies on energy optimization in different IoT applications, it is important to develop an energy-aware approach that minimizes energy consumption during the broker switching process. This can involve leveraging edge computing, optimizing resource utilization, and implementing energy-efficient algorithms and protocols.

In summary, energy consumption in IoT is a critical factor to consider when investigating the implementation challenges of a multibroker MQTT platform for smart environmental monitoring. By drawing insights from the literature on energy optimization in various IoT domains, it is possible to develop strategies that minimize energy consumption during broker switching, contributing to the overall efficiency and sustainability of the IoT system.

2.6.3. Message Loss in IoT and Its Relevance to Seamless Connectivity

In the context of seamless connectivity and broker switching in dynamic IoT environments, the issue of message loss holds significant relevance. The reliable and uninterrupted delivery of messages is crucial for maintaining an optimal user experience and system efficiency. This section explores existing research on message loss in IoT and its connection to the implementation challenges of a multibroker MQTT platform for smart environmental monitoring.

One approach to address message loss is through the utilization of osmotic computing principles, as highlighted in [57]. By creating a distributed network of brokers that dynamically diffuse to edge resources when needed, proximity is prioritized. This architecture optimizes responsiveness and reduces latency by deploying new brokers closer to devices. Such an approach aligns with the objective of maintaining continuous and reliable connections to the nearest broker in dynamic IoT environments with mobile users. Wireless Sensor Networks (WSNs) face challenges such as packet loss and end-to-end delays, as discussed in [58]. The proposed gateway-to-gateway load balancing solution aims to reduce end-to-end delay and improve packet delivery rate in a static-defined WSN cluster network. Although the study focuses on static networks, the concept of load balancing can be relevant to broker switching scenarios. By dynamically selecting the appropriate broker based on device

proximity and network conditions, it may be possible to minimize message loss and improve overall performance.

Another research study by Khare et al. [59] specifically addresses message loss in IoT environments caused by a high number of devices connected to a server. The proposed solution integrates an Identity (ID) technique into the Message Queuing Telemetry Transport Protocol (MQTT) push message service. By combining a hexadecimal number with the actual message, data retrieval in edge devices becomes possible in case of message loss. This approach demonstrates efficient transfer of the first message and satisfactory latency, aligning with the goal of minimizing message loss during broker switching.

Enhancing the robustness of MQTT protocol in mobile IoT networks is the focus of [60]. The proposed architecture integrates a Disruption Tolerant Network (DTN) approach to address the limitations of MQTT in handling device mobility and disruptions in connections. Real device experiments validate the feasibility of this architecture, showcasing low round-trip time and message loss percentages in different network scenarios. This research provides insights into addressing message loss in mobile IoT environments, which is crucial when users switch brokers while on the move.

Furthermore, [61] explores connectivity strategies for container tracking near coastlines, but the concepts discussed can be extended to broker switching scenarios. The relay-aided schemes using vessels or unmanned aerial vehicles (UAVs) as intermediate nodes aim to mitigate message loss and improve system performance by distributing transmission requests and reducing contention. This indicates the potential benefits of using relays or intermediate nodes to minimize message loss during broker switching in dynamic IoT environments.

By analyzing the existing literature on message loss in IoT and its connection to broker switching, it becomes evident that addressing message loss is a critical aspect of ensuring seamless connectivity in dynamic IoT environments. The identified studies provide insights into load balancing techniques, ID integration, DTN approaches, and relay-aided schemes, which can be considered when investigating the implementation challenges of a multibroker MQTT platform for smart environmental monitoring.

2.7. MQTT Protocol

In the context of seamless connectivity and investigating the implementation challenges of a multibroker MQTT platform for smart environmental monitoring, the MQTT protocol plays a critical role in facilitating efficient and reliable communication between sensors and brokers. This section explores relevant research studies that examine various aspects of the MQTT protocol, shedding light on its performance and suitability for dynamic IoT environments.

One approach presented by the author [62] focuses on addressing the interconnection challenges of embedded systems in networks using multi-agent systems. This approach dynamically models and creates links between MQTT brokers, aiming to enhance the efficiency and effectiveness of dynamic interaction management in IoT networks. The results demonstrate that the proposed approach outperforms the initial model, achieving 20% higher efficiency and significant advancements in dynamic

interaction management. However, further evaluation and adjustment of certain parameters $(\alpha \; and \; \beta)$ are required to fully meet the constraints of high connectivity, which aligns with the objectives of this research.

In a comparative study conducted by Koziolek et al. [63], three distributed MQTT brokers (EMQX, HiveMQ, and VerneMQ) are evaluated in an enterprise IoT scenario on an edge gateway cluster. The evaluation covers performance, resilience, scalability, security, and usability. EMQX demonstrates the highest performance with 28K messages per second, while HiveMQ achieves no message loss in the test scenario. VerneMQ, an open-source solution, offers similar features to commercial brokers. The evaluation employs a systematic approach using the Goal/Question/Metric scheme and deploys the brokers on redundant edge gateway servers with the StarlingX platform. This research provides valuable insights into the performance and characteristics of different MQTT brokers, which can inform the selection and configuration of brokers in the multibroker MQTT platform under investigation.

Luzuriaga et al. [64] focus on evaluating the performance of the MQTT protocol, along with the AMQP protocol, under mobile or unstable wireless network conditions. The study assesses whether these protocols can provide satisfactory service based on the load requirements of applications, considering factors such as message size and communication rates. They utilize a synthetic load generator called amqperf to send messages with sequence numbers and detect losses or messages delivered in different orders. The evaluation also includes measuring the impact of a mobile producer/publisher switching between Wi-Fi access points in the same IP network. The findings reveal the performance characteristics of the protocols during network transitions, with mean jitter values oscillating between 3 and 6 seconds, and peaks of up to 7 seconds observed for high transmission rates of 100 messages per second. These insights highlight the importance of investigating the impact of broker switching on latency and message loss in dynamic IoT environments, aligning with the objectives of this research.

By incorporating these relevant studies, this section establishes a foundation for understanding the MQTT protocol's role in seamless connectivity within a multibroker MQTT platform for smart environmental monitoring. The insights gained from these studies will contribute to identifying potential improvements and solutions to minimize latency and message loss during broker switching, as well as inform the evaluation of the proposed improvements and solutions in subsequent sections of this research.

2.8. Broker-Based Communication Systems

Broker-based communication systems are a type of distributed system [65] where a software broker facilitates communication between various application components. In this system, the broker acts as a middleman, handling the communication complexities of a distributed system such as message routing, request-reply correlation, encoding and decoding, and ensuring reliable and secure transmission of messages. This communication architecture aims to abstract the underlying complexities of inter-process communication in a distributed environment, thus enabling application developers to focus more on their application logic instead of communication details.

There are multiple advantages to broker-based communication systems. It promotes decoupling of services, as they only need to know about the broker, not each other, enhancing modularity and scalability. The broker can provide additional services such as data transformation, transaction management, load balancing, and fault tolerance. However, broker-based systems have certain drawbacks as well. The broker has the potential to become both a single point of failure and a performance bottleneck, and also introduce latency due to the extra layer of communication. In addition, deploying and managing the broker infrastructure could be complex and cost-intensive.

Despite these challenges, broker-based communication is widely adopted in various types of systems, such as event-driven systems, message-oriented middleware, and service-oriented architecture (SOA). For instance, Apache Kafka, RabbitMQ, and Google Pub/Sub are all examples of broker-based systems, each with their own characteristics and optimizations to handle the challenges. Thus, broker-based systems are an important design pattern in distributed systems and continue to be relevant with the proliferation of cloud computing and microservices architecture.

2.9. Prior Research on Broker Switching, Latency Calculation, and Message Loss

Connecting diverse devices, sensors, and real-time applications is the main problem of IoT automation. Recent developments for a wide range of IoT ecosystem solutions and devices have been shown through research. However, each compatible technology has its own exclusive communication infrastructure and method, which causes an IoT interoperability problem. To solve this study, the development of a bridge configuration for IoT interoperability utilizing the MQTT and CoAP protocols is covered in the article. In order to provide smooth communication between MQTT and CoAP, the authors suggest using the Ponte message broker as a bridge. They go through the Ponte message broker's architecture, configuration procedures, and parts. Comparing delay and message delivery rates in experimental findings shows how well it supports interoperability. In IoT contexts, the Ponte bridge configuration offers easy communication between MQTT and CoAP devices. The article addresses prospective applications and emphasizes the need for interoperability in IoT systems. Overall, it offers insights into a workable strategy for facilitating communication across IoT devices that use various protocols[66].

In this study, application layer protocols for edge-centric real-time IoVT applications are evaluated quantitatively. The study, which was conducted during the UEMCON conference, emphasizes the need to choose appropriate protocols for effective and trustworthy communication. Throughput, dependability, and other performance parameters are measured and examined using the evaluation technique. Based on their experimental findings, protocols are contrasted, highlighting their advantages and disadvantages. V2V and V2I communications are included in the evaluated scenarios. The article offers insights into protocol features, benefits, and use cases to help readers choose the best protocol for their particular IoVT application needs. Overall, it provides useful knowledge for industry professionals and scholars [67].

The MQTT-ST protocol is described in the article as a means of scalable and dependable communication between dispersed MQTT brokers. The protocol makes use of a spanning tree topology for effective broker coordination and message routing. The architecture, message format, and algorithms of the protocol are explained in great detail. Results from simulations show that MQTT-ST is superior to competing protocols in terms of latency, network utilization, and load balancing. The authors talk about its application in many situations, highlighting its potential to improve performance and dependability in IoT and smart city installations. Overall, the MQTT-ST protocol addresses the demand for increased scalability and reliability in distributed MQTT broker networks by providing useful insights and advantages to researchers and practitioners [68]. For the drone nodes that are actively functioning as edge computing components, a message transmission method has been presented in this study. A drone node known as "EdgeDrone" has been given the new designation. Standard message transmission protocols like MQTT and MQTT-SN do not provide the greatest efficiency due to the node's dynamism. The ”spray and focus” opportunistic routing method is a combination with our upgraded MQTT and MQTT-SN protocol solutions. It continues to monitor transitivity and encounter history over time. Further testing of the new technique on a real-world test bed revealed much better performance in terms of latency, publisher bandwidth, and operating time. The findings also showed that the upgraded MQTT-SN mechanism performs well in terms of memory use and energy dissipation. Additionally, the standard and improved mechanisms have been compared in the simulation platform, and it has been shown that the enhanced MQTT receives fewer acknowledgments when nodes are sporadically connected. And the same is true for MQTT-SN. Given that the number of acknowledgments is inversely correlated with the number of messages lost, it is clear why the improved approach performs better in the problematic situation[69].

3. SYSTEM MODEL AND METHODOLOGY

3.1. Methodology

This section describes the methodology used to evaluate network latency and switching latency in dynamic IoT environments. The assumptions and constraints considered during the research process are outlined in Section 3.1.5. The metrics used to evaluate the performance of the system are defined in Section 3.1.6. The research methodology adopted for this study is given below:

3.1.1. Case Study 1: Cloud-Based Multi-Broker Deployment

- Brokers are deployed on virtual machines (VMs) in a central cloud provider.
- Sensors publish data to the brokers using the wide area network (WAN).
- Clients receive data from the brokers using the WAN as well.

This case study represents a traditional cloud computing environment. As shown in figure 1, the brokers are located in a central location, and the sensors and clients are connected to the brokers over the WAN. This architecture has several advantages, including: Scalability: The brokers can be scaled up or down to meet the needs of the application. Reliability: The brokers are located in a central location, which makes them more reliable than edge devices. Security: The brokers are protected by the cloud provider's security measures. However, this architecture also has some disadvantages, including: Latency: The data must travel over the WAN to reach the brokers, which can introduce latency. Cost: The cloud provider charges for the resources used by the brokers.

3.1.2. Case Study 2: Local Area Network(LAN)-Based Multi-Broker Deployment

- Brokers are deployed on edge.
- Sensors publish data using the local area network (LAN).
- Clients receive data within the same LAN environment.

This case study represents a local area network (LAN)-based edge computing environment. As shown in figure 2, the brokers are located on the edge, which are connected to the sensors and clients over the LAN. This architecture has several advantages, including: Low latency: The data does not have to travel over the WAN, which reduces latency. Cost-effectiveness: The edge devices are generally less expensive than cloud-based brokers. Flexibility: The brokers can be moved easily, which makes them well-suited for mobile applications. However, this architecture also has some disadvantages, including: Security: The brokers may not be much secured as cloud.

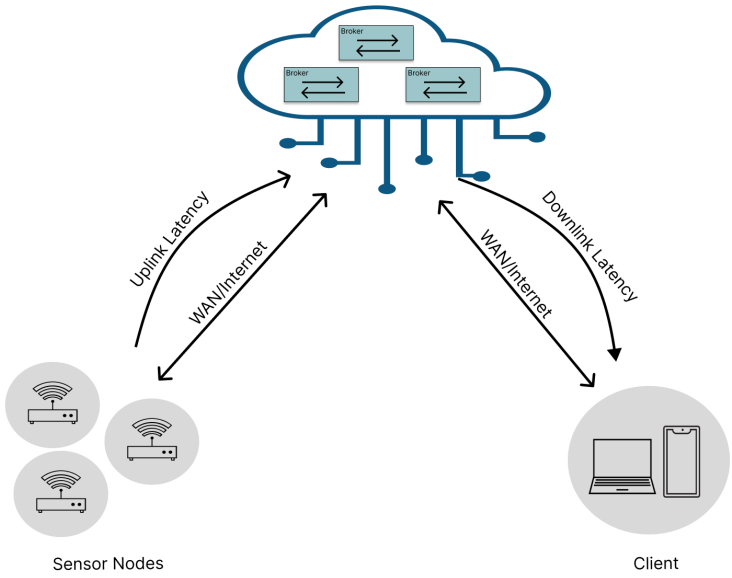

Figure 1. Cloud-Based Multi-Broker Deployment

3.1.3. Case Study 3:Multi-Layer LAN Network Based Multi-Broker Deployment

- Brokers are deployed on edge devices.
- Publishers are connected to layer 2, and subscribers are connected to layer 1 within a same network (LAN) as shown in fig: 3.

This scenario simulates an edge computing environment. This case study represents a multi-layer network-based edge computing environment. As shown in figure 3, the brokers are located on the edge, which are connected to the sensors and clients over a multi-layer separation in network. This architecture has several advantages, including: Compliance: The data can be kept within the organization's control, which can help to meet compliance requirements. Cost-effectiveness: The edge devices can be less expensive than cloud-based brokers. However, this architecture also has some disadvantages, including: Complexity: This multi-layered network must be set up and managed, which can be complex in administrative perspective.

3.1.4. Fundamental Latency in Each Scenario

In a wide area network (WAN), the overall system latency is the delay in the propagation of a message from the publisher network to the client. This delay includes the following mentioned latencies, as shown in figure 4:

- Publisher network latency (La): This is the delay in the publisher network, which is the network that the publisher is connected to. This delay can be caused by factors such as the distance between the publisher and the broker, the bandwidth of the network, and the number of devices on the network.

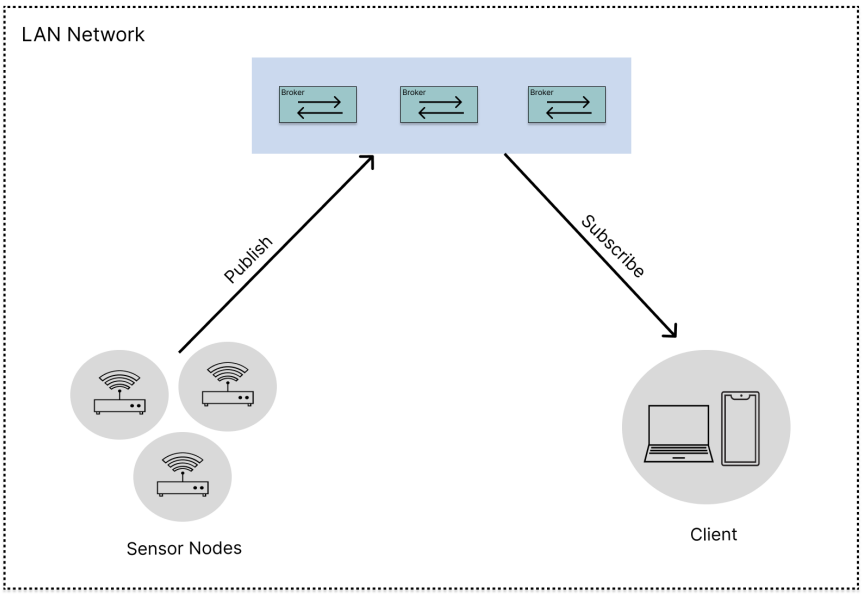

Figure 2. Local Area Network(LAN)-Based Multi-Broker Deployment

- Internet latency to the main server (Lb): This is the delay in the internet, which is the network that the broker is connected to. This delay can be caused by factors such as the distance between the broker and the main server, the bandwidth of the internet, and the number of devices on the internet.
- Central server latency (Lc): This is the delay in the central server, which is the server that receives messages from the broker and sends them to the clients. This delay can be caused by factors such as the distance between the broker and the central server, the bandwidth of the network, and the number of devices on the network.
- Transmission latency back to the client (Ld): This is the delay in the transmission of the message back to the client. This delay can be caused by factors such as the distance between the central server and the client, the bandwidth of the network, and the number of devices on the network.
- Client network latency (Le): This is the delay in the client network, which is the network that the client is connected to. This delay can be caused by factors such as the distance between the client and the broker, the bandwidth of the network, and the number of devices on the network.

In case 1, the total latency of the cloud network 1 as mentioned in section 3.1.1 is calculated as follows.

$L = La + Lb + Lc + Ld + Le$

In case 2 as mentioned in section: 3.1.2 and shown in figure 2, the WAN latency is eliminated. In this case, the total latency (L) is calculated as follows:

$L = La + Lc + Ld + Le$

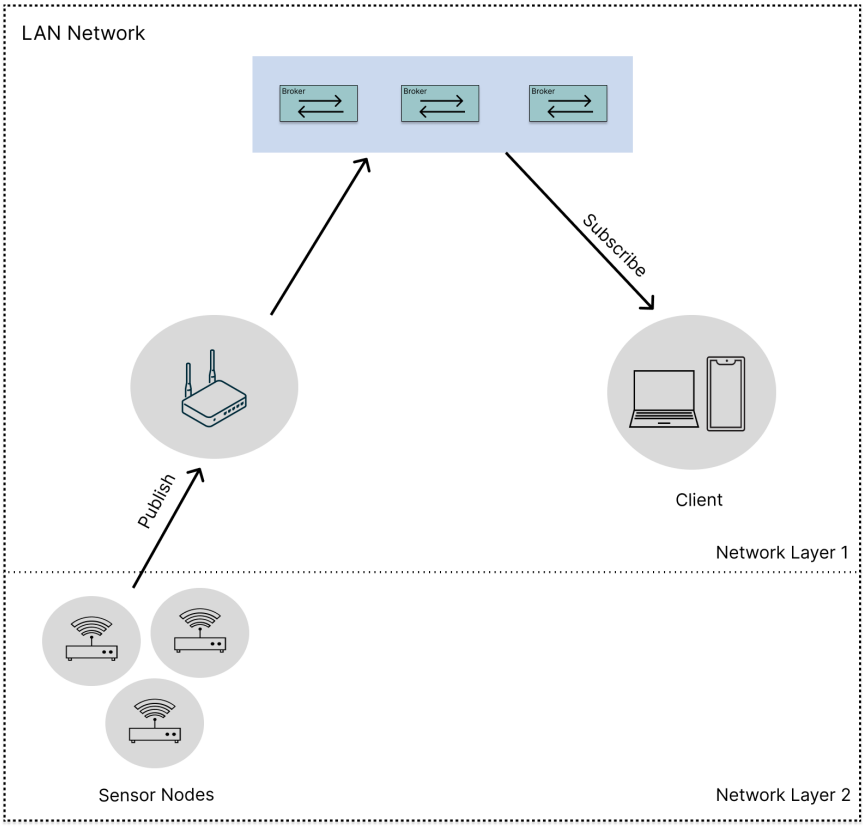

Figure 3. Multi-Layer LAN Network Based Multi-Broker Deployment

In case 3 as mentioned in section: 3.1.3, the total latency of the network depends on the number of layers introduced between the publisher, broker, and client. For the test scenario as shown in 3, the total latency is:

$L = la' + lc + ld + le$

Where la' represents the total latency of all layers introduced between the publisher and broker.

As shown in the figure below, Lc' and Ld' and Le' represents the total complexity in terms of latency for the testing scenarios as

Based on the above test cases, the methodology to eliminate the dynamic variables of latency is optimized. It is assumed some constant factors are mentioned in the section "Assumptions and Constraints" 3.1.5 section and developed performance evaluation metrics.

3.1.5. Assumptions and Constraints

The following assumptions and constraints are made in this thesis to finalize the metrics of performance evolution given in next section “Metrics of Evaluations 3.1.6”: **The sensors are always connected to the network**: This means that the sensors must have a reliable internet connection. If the sensors are not connected to the network, they will not be able to publish data. **The brokers are always available:** This means that the brokers must be up and running and able to receive messages from the sensors.

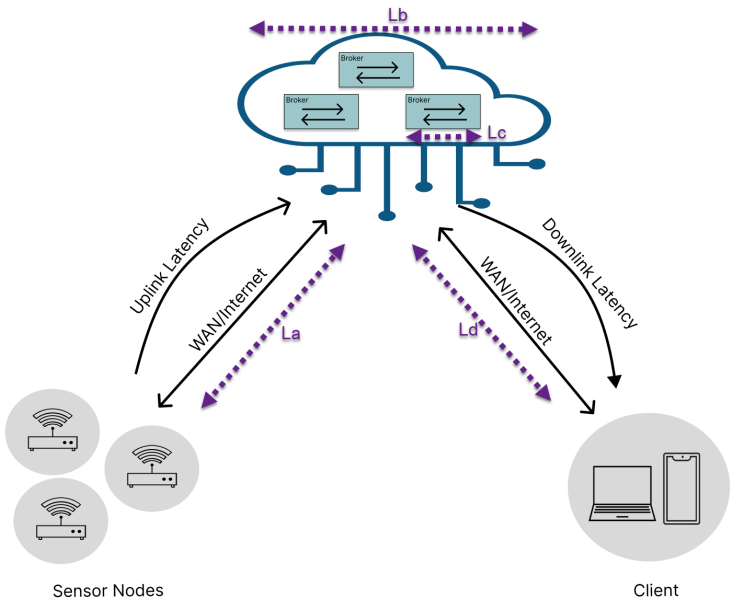

Figure 4. Fundamental Latency

If the brokers are not available, the sensors will not be able to publish data. **The clients are always available:** the clients must be up and running and able to receive messages from the brokers. If the clients are not available, the brokers will not be able to send messages to the clients. **The network bandwidth is sufficient to support the data traffic** the network must be able to handle the amount of data that is being published and consumed. If the network bandwidth is not sufficient, the data will not be able to be published or consumed in a timely manner. **The latency of the network is constant** the time it takes for a message to travel from the sensor to the broker and then to the client is always the same. If the network latency is not constant, the time it takes for a message to travel will vary, which can affect the performance of the system. **Time drift in clock synchronization between two consecutive messages at publisher or client is negligible** because time drift is synchronized at client side using gradient time synchronization method [70]

3.1.6. Metrics for Performance Evaluation

The following metrics are used to evaluate the performance of the system in each scenario:
Network latency: The time it takes for a message to travel from a sensor to a client. This metric is important because it measures how quickly the system can respond to events.
Switching latency: The time it takes for a client to switch from one broker to another. This metric is important because it measures how quickly the system can adapt to changes in the network topology. In general, lower latency is better. However, the specific requirements for each scenario will vary. For example, a system that is used

for real-time applications will need to have lower latency than a system that is used for batch processing. Here are some additional details about each metric:

Network latency is affected by a number of factors, including the distance between the sensor and the client, the bandwidth of the network, and the type of network.

Switching latency is affected by a number of factors, including the number of brokers in the system, the bandwidth of the network between the brokers, and the type of switches. By measuring these metrics, you can identify areas where the system can be improved. For example, if the network latency is high, you may want to consider using a different network or increasing the bandwidth. If the switching latency is high, you may want to consider using a different type of switch or reducing the number of brokers in the system.

3.1.7. Implementation Approach

The research methodology adopted for this study consists of the following steps:

System Implementation: The system model is implemented in a simulation environment that represents the different deployment scenarios. This includes deploying brokers on virtual machines (VMs) in a central cloud provider for case study 1 (section: 3.1.1), on edge devices for case study 2 (section: 3.1.2), on the multi-layer network on edge devices for case study 3 (section: 3.1.3)

Data Collection: The simulation is run with various parameters to collect data for evaluation. This includes gathering data on network latency and switching latency in each scenario.

Data Analysis: The collected data is analyzed to understand the variations and trends in network latency and switching latency. The analysis aims to answer research questions related mentioned in section: 1.2.2 (2a2b)

3.1.8. Optimized Methodology to Calculate Latency Calculation

This study's approach is aimed at identifying and addressing the key elements contributing to network latency, which are elaborated on in the "Fundamental Latency in Each Scenario" section. For precise latency measurement, the procedure includes coordinating the client's clock with the clocks of the nodes. This coordination process aids in mitigating any effects of clock drift on latency measurements. To ensure accuracy over longer periods, clock synchronization is carried out regularly.

$$TL = ToMR - ToMS \tag{1}$$

where
TL = Total Latency
ToMR = Time of Message Receipt (Client)
ToMS = Time of Message Sending (Synced with Client's Time)

Once the clocks are aligned and the message sending time is coordinated with the client's clock, the client determines the latency by calculating the difference between the time of receipt and the time of sending. This approach depends on the sender including a timestamp from the device in every message, enabling the client to precisely determine the network latency. Switching latency, in contrast, emphasizes the time duration from data receipt after the connection is transferred to the next broker, making sure no messages are misplaced in the shift. These formulas yield a numerical representation of the latency experienced within the system, which aids in the assessment of the performance and efficiency of the MQTT-based infrastructure concerning latency.

4. SIMULATION AND EXPERIMENTAL SETUP

4.1. System Description

The system model used for evaluating network latency and switching latency consists of multiple brokers deployed in different scenarios, sensors for data publication, and clients for data reception. The brokers are responsible for receiving data from the sensors and forwarding it to the clients. The sensors are responsible for collecting data from the environment and publishing it to the brokers. The clients are responsible for receiving data from the brokers and displaying it to the user.

4.1.1. Hardware Architecture:

A variety of sensors were selected based on the data type. Some sensors require more bandwidth than others, such as sound sensors. Others may need priority at the time of an event, such as fire or temperature sensors. Toxic gases may also require additional priority due to their odorless nature and the need for urgent action. Our designed hardware enables us to test in-depth scenarios based on requirements. The sensors that were used in the study were:

- MQ135 Air Quality Detector Sensor
- KY-038 LM393 Sound Detection Module
- DHT11 Temperature and Humidity Sensor Module
- KY-018 LDR light sensor module

The sensors were connected to a custom-designed PCB (Printed Circuit Board) that allowed them to be connected to the network and publish their data to the brokers. The PCBs were manufactured using LPKF CircuitPro PM software and fabricated in a Fablab using a machine name lpkf protomat s62 (machine). Soldering was performed on the boards to connect a sensor on the board. Three such PCBs were placed at different locations to collect sensor data. the PCB Schematics was designed using Altium Designer software. The software allows users to create a schematic of the circuit and then generate a PCB layout. The schematic is a diagram of the circuit, showing the components and their connections. The PCB layout is a physical representation of the circuit, showing the components and their connections on the PCB. The following steps were used to design the PCB: The sensors were identified and their specifications were gathered.

1. The components that would be needed to connect the sensors to the network were identified.
2. The schematic of the circuit was created as shown in figure. 5. This PCB is responsible for gathering data from sensors and transmitting it to the broker using the MQTT protocol.
3. The PCB layout was generated as shown in figure. 6

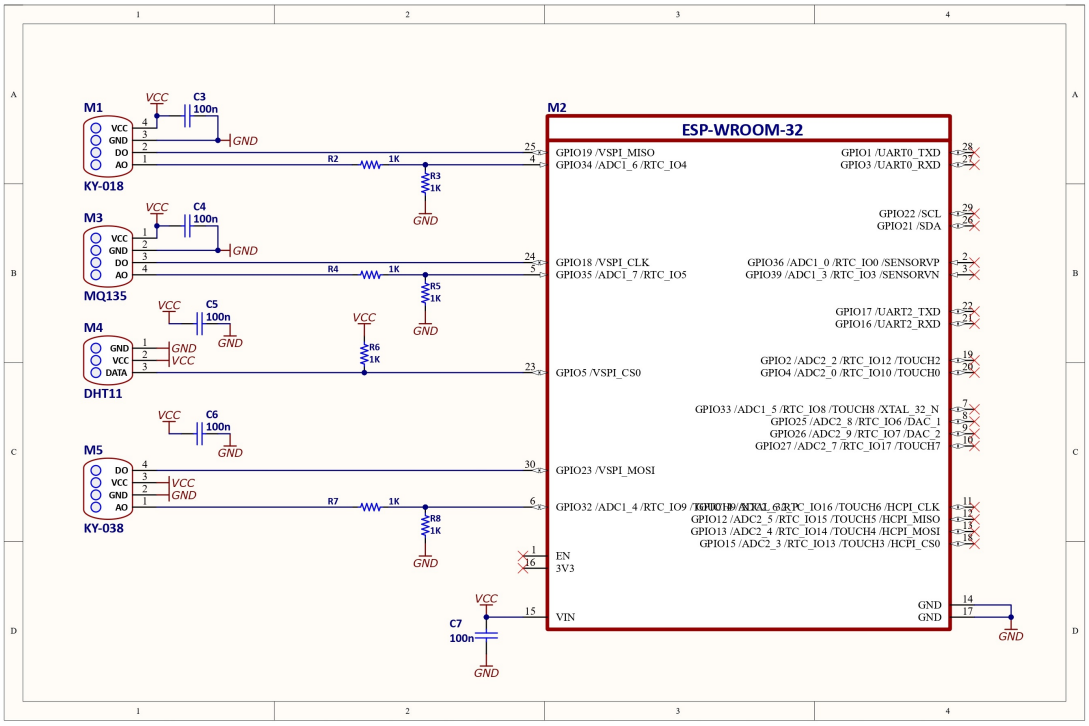

Figure 5. Hardware Schematics

4. The PCB was fabricated in a Fablab using an LPKF protomat 62 machine.

5. Soldering was performed on the boards to establish the necessary connections. The figure. 7 shows the actual appearance of working PCB.

The system model is divided into three scenarios as described in section methodology 3.1

4.2. Software Architecture

Problem Statement: As already discussed, the goal of this research thesis, titled "Seamless Connectivity: Investigating Implementation Challenges of Multibroker MQTT Platform for Smart Environmental Monitoring," is to explore the challenges associated with building a reliable and scalable MQTT-based infrastructure for smart environmental monitoring systems. One of the key challenges this study focuses on is maintaining seamless connectivity and message reliability across multi-brokers.

Solution Overview: In response to the challenges identified, this study presents a proposed solution that involves the configuration of multi-MQTT brokers and the utilization of bridging mechanisms. The solution aims to facilitate seamless message publishing and reception across brokers, ensuring reliable and low-latency communication. Moreover, the implementation of a distributed broker allows for studying its implementation and achieving the associated benefits. By distributing the data, it is broadcasted to all brokers, ensuring its availability everywhere. This

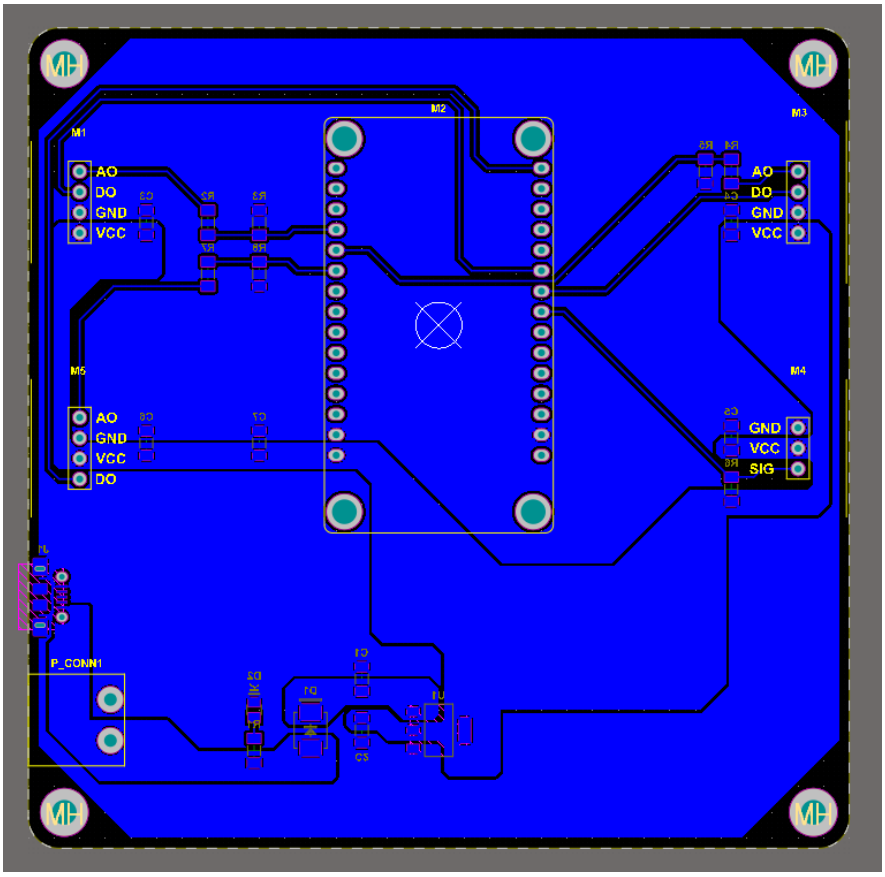

Figure 6. PCB layout

approach ensures that the data is accessible and present on each broker, enhancing the overall availability and reliability of the system. Additionally, the system incorporates essential features such as message tracking and latency calculation to effectively monitor and evaluate its performance.

Solution Components:

1. **Docker Compose Configuration:** ‘docker-compose.yml‘ file contains the configuration for setting up three MQTT brokers, namely broker1, broker2, and broker3. Each broker is configured with persistence and session storage for maintaining message state and connection information.

2. **Broker Configurations:** The configurations for each broker are defined in the mosquitto.conf files located within the respective directories of config/broker1, config/broker2, and config/broker3. These configurations define the bridge connections between brokers and other essential settings.

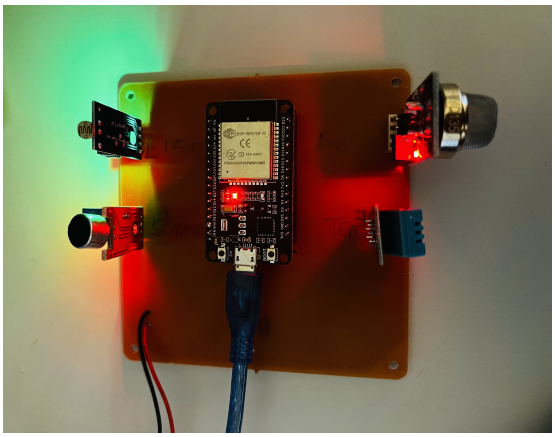

Figure 7. PCB

Algorithm 1 Mosquitto Docker Configuration Pseudo-code

```
Set persistence as true
Set persistence_location as '/mosquitto/session_storage/'
Initialize listener at port 1883
Allow anonymous connection as true
Set pid_file as '/mosquitto/session_storage/mosquitto.pid'
Set log destination as a file '/mosquitto/session_storage/mosquitto.log'
Set log timestamp format as '%Y-%m-%dT%H:%M:%S'
Start connection 'bridge-2'
    Set address as 'broker2:1883'
    Register to topics with QoS and set publish and subscribe permissions
    Set try_private as true
End connection
Start connection 'bridge-3'
    Set address as 'broker3:1883'
    Register to topics with QoS and set publish and subscribe permissions
End connection
```

3. **Publisher Script:** The ‘publisher.ino‘ script deployed on ESP32 publishes messages to the MQTT network. It generates message payloads in JSON format, including a unique tracking ID for each message. The script publishes messages regularly.

Algorithm 2 Publisher Script Pseudo-code

```
procedure BEGIN
    Initialize WiFi with given SSID and password
    Initialize DHT sensor
    Set MQTT server for each client
    Connect each client to its respective MQTT broker
end procedure

procedure LOOP
if [current time - last message time > 1 second] then
    Read values from all sensors
    Generate a random UUID for each message

    for each sensor reading do
        Construct a JSON payload including the sensor value,
        generation time, and tracking ID
        Publish the payload to its respective topic on each MQTT broker
    end
    end
end procedure
```

4. **Receiver Script:** The ‘receiver.py‘ script subscribes to the MQTT network and receives messages published by the ‘publisher.ino‘ script. It ensures that messages are processed only once by maintaining a tracking mechanism using a file to store received message IDs. The script calculates the message reception time, latency, and skips duplicate messages.

Algorithm 3 Pseudocode for receiver:

```
Import necessary libraries
Do initializations: Set Broker Host, Port, and Topics
Create an MQTT client
Set the on_connect and on_message functions as the client's callback functions
Connect to the broker
Start the MQTT client loop
Create a database connection and return it
procedure ON_CONNECT
    If connection is successful, print a success message and subscribe to topics
    If connection fails, print an error message
end procedure
procedure ON_MESSAGE
    Decode the message payload
    Try to parse the JSON payload
    If successful, extract the sensor value, generation time, and tracking ID
    If first iteration, sync the published device time with receiving device time
    If the tracking ID is not in the tracking IDs file, add it and write the updated
tracking IDs to the file
    If the message, generation time, and tracking ID are valid:
    Calculate the latency and execution time
    Insert the data into the database
end procedure
```

How the Solution Works:

1. The MQTT brokers are configured to bridge connections between each other using the provided ‘mosquitto.conf‘ files. This enables message propagation across brokers.

2. The ‘publisher.ino‘ script publishes sensors measurements with unique tracking IDs and publishes them to the MQTT network. It ensures that messages expire after a certain duration (2 minutes) to maintain message freshness.

3. The ‘receiver.py‘ script subscribes to the MQTT network and receives published messages. It maintains a tracking mechanism to identify and skip duplicate messages based on the received tracking IDs. The script calculates message reception time, and latency, and prints the relevant information.

4.3. Hardware-Software Integration

The deployment of a broker on the edge for hypothesis validation requires careful consideration of several factors, including clock synchronization, queuing delay, packet loss and re-transmission, and jitters. One of the most important factors is clock synchronization. In a distributed system, accurate latency measurements depend on synchronized clocks. Therefore, a reliable clock synchronization protocol must be

and has been implemented across the edge devices and the broker. This will ensure that the clocks of all devices are closely aligned, minimizing discrepancies that could introduce errors in latency calculations. As a result, consistent and accurate latency measurements can be obtained, which is essential for accurate hypothesis testing.

Considering these factors involved in deploying a broker on the edge and implementing corresponding optimizations, hypothesis testing is conducted with improved accuracy and credibility. This approach creates a controlled and reliable environment that minimizes potential sources of error, ensuring trustworthy results for the hypothesis being tested on edge devices.

5. RESULTS AND ANALYSIS

5.1. Test Case Analysis

As discussed earlier (see equation 1), to calculate the overall latency, the round trip time is measured. This involves capturing the time it takes for the message to travel from the device node (publisher) to the broker and then forward it to the client (subscriber). By considering the complete round trip, the total latency experienced by the packet is accurately assessed.

Furthermore, to ensure reliability, these experiments involve generating a series of messages and recording the time it takes for each message to complete its round trip. By averaging these measurements, a representative value for the overall latency is obtained.

The results of all three use cases are presented separately below, along with a summary of their core relevance at the end of this section.

5.1.1. Test Case: Cloud-Based Multi-Broker Deployment

As discussed earlier, for a cloud-based scenario (see section 3.1.1), the latency of the entire packet's journey is calculated first. This latency measurement is a reference point for determining the optimal placement of the broker, which in this case is determined to be at the cloud. The analysis of results are described below.
Figure 8 shows that the average latency over the cloud network is approximately 91 milliseconds, with significant jitters. These jitters are primarily attributed to the overall Quality of Service (QoS) of the Internet. Here Quality of Service refers to the capability of a network to provide different levels of performance and reliability for different types of traffic or network services.

In this case, the observed jitters are disrupting the latency, indicating that the Quality of Service on the cloud network may not be consistent or reliable. This observation reinforces the need to deploy an MQTT broker on the edge, which can help mitigate the effects of network jitters and ensure a more stable and reliable communication environment.

Moreover, the highest observed peak jitter on the cloud network is 495.1 milliseconds. And the average switching latency on the cloud is recorded to be 47.6 milliseconds. A comparison of these values is provided later in Analytics section 5.2

5.1.2. Test Case: LAN-Based Multi-Broker Deployment"

The second test case evaluates the performance of an MQTT broker deployment in a local area network (LAN) (See section 3.1.2. This configuration aims to minimize network congestion, which is typically associated with cloud-based deployments.

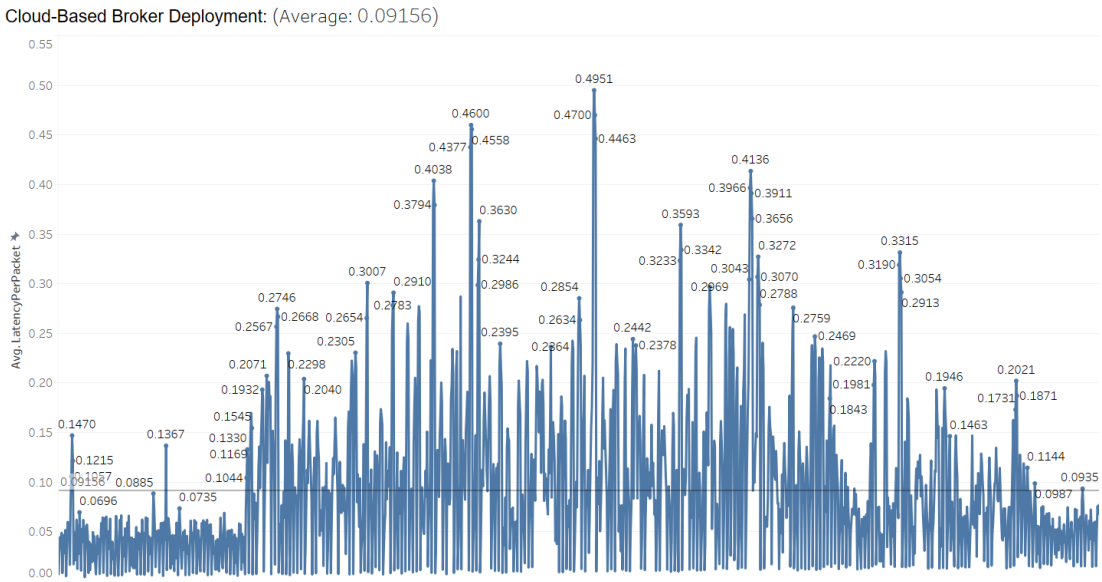

Figure 8. Cloud-Based Multi-Broker Deployment

In Figure 9, a notable decrease in jitters is observed on the edge network over LAN, resulting in an overall reduction of latency by up to 10.85ms. The maximum jitter observed on the LAN network is approximately 183.4 ms, which is significantly lower than the peak latency observed on the cloud network. This reduction in jitter effectively contributes to a lower overall latency in the network. Similar improvements are also observed in the switching latency, with a recorded switching latency of 6.8ms on the edge network.

By connecting the client and the broker closely within the LAN, the communication distance is reduced. This proximity enables faster transmission and reception of messages, leading to lower latency. As the data does not need to traverse external networks, such as the internet or cloud infrastructure, potential bottlenecks, and congestion are minimized.

However, it is important to consider that the overall latency of the system can still be influenced by the data load present on the local network. Heavy data traffic or congestion within the local network can introduce delays and increase latency.

Additionally, in this test case, the measurement of switching latency (definition: 3.1.6 remains relevant. Switching latency includes activities such as connection establishment, disconnection from the previous broker, and protocol handshakes. Measuring switching latency provides insights into the efficiency of the broker switching process and its impact on overall system performance within the LAN environment.

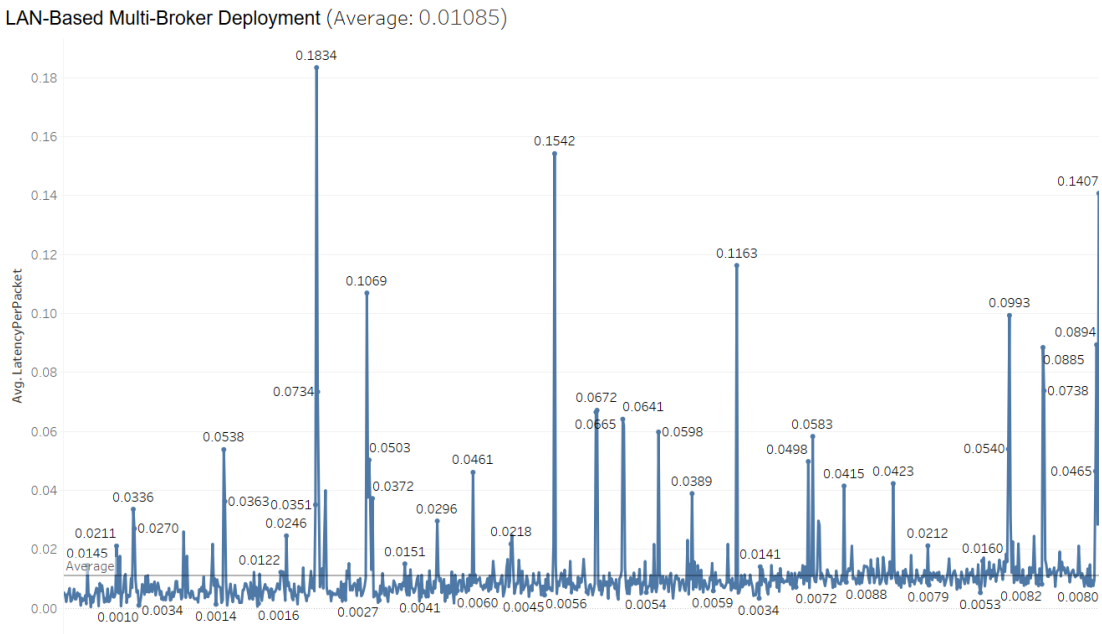

Figure 9. LAN-Based Multi-Broker Deployment

5.1.3. Test Case: Multi-Layer LAN Network Based Multi-Broker Deployment"

In some cases, it may not be possible to establish a direct connection between the client and broker at the edge due to factors such as traffic congestion, resource limitations, or network complexity. In these situations, it is necessary to introduce multiple layers of network within the local area network (LAN) (described in section 3.1.3. This intermediate placement of the broker and client within the network is an important test case to explore.

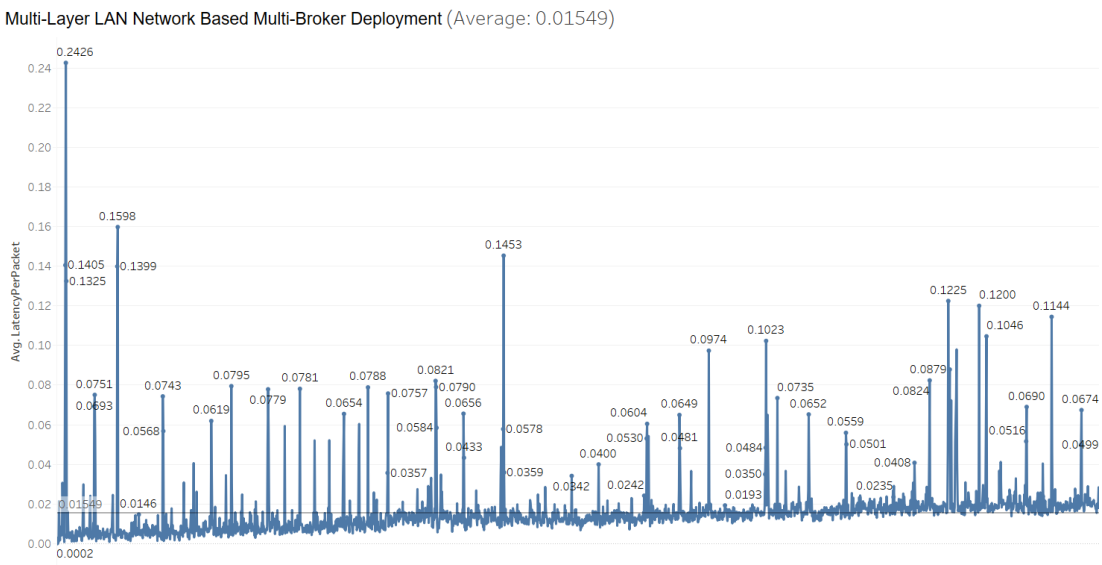

Figure 10. Multi-Layer LAN Network Based Multi-Broker Deployment

In figure 10, the introduction of an intermediate layer for testing a multi-layer

network results in an increase in the overall network latency to 15.4ms. However, the switching latency remains nearly the same at 6.7ms. Additionally, the peak latency of the network is 242.6 ms.

5.2. Examining the Analytics from the Above Tests

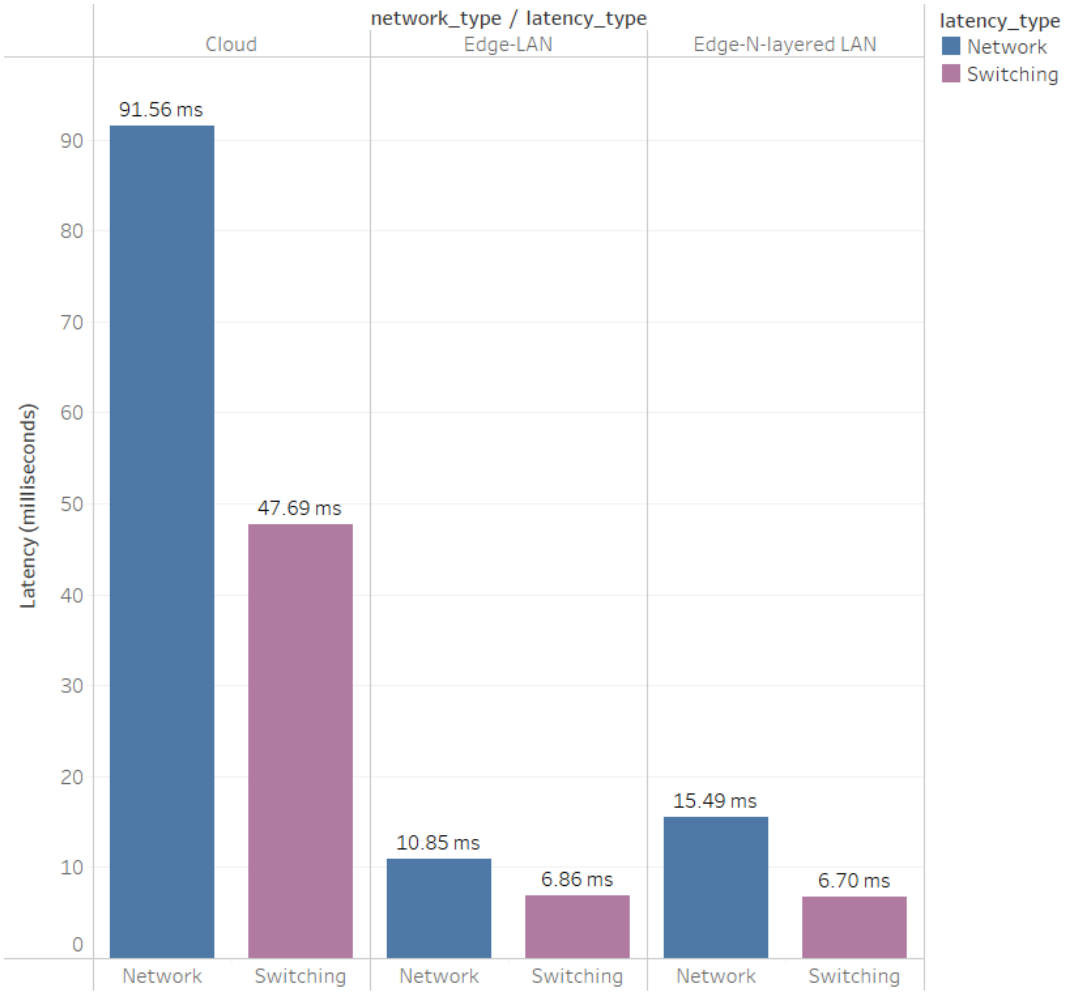

Figure 11. Switching/Network Latency

From the carried-out test, as shown in figure 11, we noted that the mean network latency in the cloud reached 91.5 milliseconds, while the edge latency averaged only 10 milliseconds. Furthermore, the mean latency in the multi-tier network structure was recorded to be 15.5 milliseconds. That means, the data shows a significant reduction in network latency at the edge, which is approximately nine times lower compared to the network latency observed in the cloud. In the specific multi-tier edge network configuration used in this test, network latency is about six times lower than the cloud, with the actual reduction depending on the number of layers involved.

We also analyzed the switching latency, which denotes the time taken for a

client to transition between different brokers. The average switching latency in the cloud is around 47 milliseconds, while it is only 6.86 milliseconds at the edge. For the multi-tier network, the average switching latency is about 6.7 milliseconds.

These results confirm the initial hypothesis and support the existing understanding of network latency. It is evident that switching latency follows the same pattern as network latency observed in traditional networks.

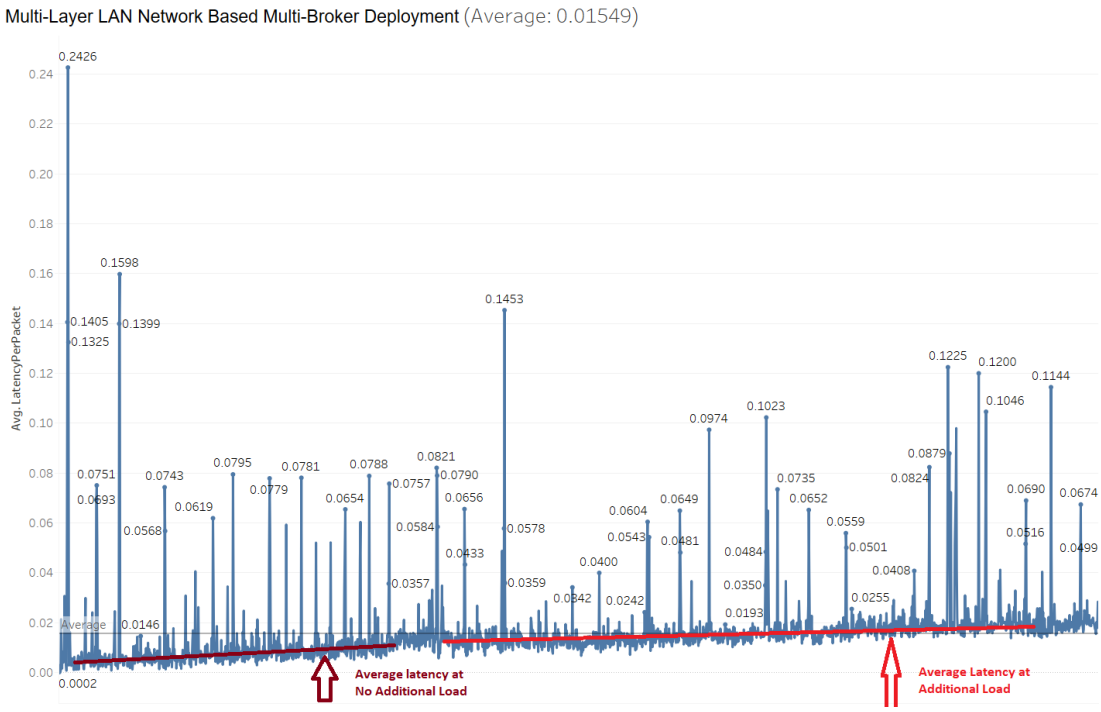

Figure 12. Multi-Layer LAN Network Based Multi-Broker Deployment - Load

Moreover, in Figure 12, it can be observed that the latency at the edge system is influenced by the current processing workload. As more tasks were performed on the edge system, the average latency increased accordingly, as indicated by the red arrows on the chart.

Interestingly, the fluctuation in latency, or jitter, experienced at the edge is significantly lower compared to that in the cloud, resulting in improved network quality of service at the edge. From the average results, it was observed that if the end-user establishes a direct connection to the MQTT broker at the edge, it further reduces network latency at the edge. This indicates the importance of finding a balance between the processing load and the client's connectivity to the edge. To explore this further, we made adjustments to the test setup and conducted preliminary validations of the hypothesis concerning direct client-to-edge MQTT broker connections using appropriate tools. These findings present opportunities for future investigations, as discussed in the Future Work section 6.3. This validation is done in next section.

5.2.1. New Hypothesis Validation: Direct Client-To-Edge Connection

To validate this hypothesis, a modification is introduced in the second test case, wherein the client is connected to the edge-placed broker on the same machine 14. This configuration established a direct and efficient connection between the client and the broker, resulting in reduced latency. However, the latency between the sensor nodes and the broker remained unchanged compared to the previous test cases. This particular setup holds promise for future implementations across diverse applications. However, this statement has undergone limited validation and is currently depicted in the figure. 13. Theoretical analysis suggests that this scenario can potentially halve the latency observed in the first, second, and third cases. The initial hypothesis was supported through system latency analysis conducted using command line interface tools such as ping or traceroute. As shown in figure 13, the result of the traceroute is less than 1 ms.

```
C:\Users\Ramisha>ping 172.24.16.1

Pinging 172.24.16.1 with 32 bytes of data:
Reply from 172.24.16.1: bytes=32 time<1ms TTL=128
Reply from 172.24.16.1: bytes=32 time<1ms TTL=128
Reply from 172.24.16.1: bytes=32 time<1ms TTL=128
Reply from 172.24.16.1: bytes=32 time<1ms TTL=128

C:\Users\Ramisha>tracert 172.24.16.1

Tracing route to DESKTOP-H38EU80.mshome.net [172.24.16.1]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  DESKTOP-H38EU80.mshome.net [172.24.16.1]

Trace complete.
```

Figure 13. Validation of new hypothesis

Interpretation of Direct Client-to-Edge Connection Hypothesis

By the above-mentioned results, we can deduce that a potential reduction in network latency can be achieved by hosting both the broker and client on the same machine at the edge. However, it is crucial to acknowledge that the validity of this hypothesis relies on several system constraints, including hardware resources, processing power, operating system limitations, and data processing and storage capabilities. In order to validate this hypothesis more comprehensively, future research can be directed toward investigating service deployment on the edge, with a specific emphasis on the applications mentioned earlier. This would require thorough verification and and thoughtful consideration of the system constraints mentioned earlier.

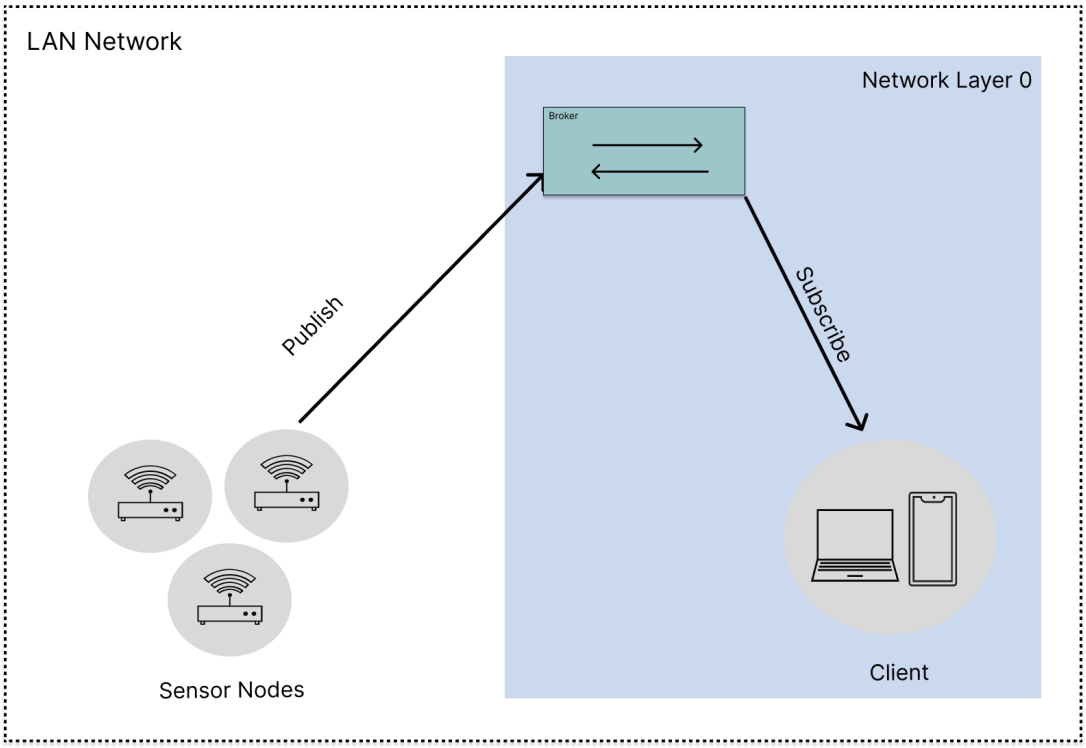

Figure 14. Emerging perspective

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

Based on the analysis of experimental results (Section 5.2), the study provides insights into research question (a) (described in section: 2a). It concludes that network latency is higher in cloud-based systems compared to edge systems like Fog and LAN, supporting the existing belief (hypothesis: 1.2.1). Additionally, the implementation of multi-brokers on edge systems significantly reduces switching latency, resulting in approximately five times lower latency compared to the cloud. This highlights the performance improvement and enhanced connectivity achieved by leveraging multiple brokers on the edge.

Based on these findings, research question (b) (in section: 1.2.2 part 2b) can be answered. It can be concluded that even with low network latency on the edge, the volume of data processing has a substantial impact on end-user applications, particularly in the domains of Virtual Reality/Augmented Reality for home or commercial users, as well as autonomous systems in industries. Additionally, there are several highly relevant applications where the distributed edge deployments of the MQTT broker, with their seamless connectivity and low latency, offer significant performance and user experience benefits. These applications include telemedicine, smart transportation, industrial automation, and environmental monitoring, as well as virtual reality (VR) and augmented reality (AR) technologies.

Moreover, VR and AR technologies heavily rely on real-time data processing and seamless connectivity to deliver immersive and interactive experiences. By leveraging the capabilities of the MQTT broker on the edge, large volumes of data, such as 3D models, sensor inputs, and real-time user interactions, can be efficiently transmitted, processed, and synchronized between VR/AR devices and edge servers. This enables enhanced user experiences with minimal latency, ensuring smooth and immersive interactions in VR gaming, training simulations, architectural visualization, and remote collaboration scenarios.

Furthermore, the low-latency and reliable connectivity offered by distributed edge deployments of the MQTT broker play a critical role in addressing the challenges faced by VR/AR applications, including real-time rendering, accurate positional tracking, and synchronized multi-user experiences. By offloading data processing and communication tasks to the edge, VR/AR applications can provide seamless and responsive experiences, improving overall performance and user satisfaction.

Distributed edge deployments of the MQTT broker possess significant potential to revolutionize VR and AR applications. By enabling efficient data transmission, processing, and synchronization, they enhance immersion, interactivity, and the real-time nature of these experiences, opening doors to innovative applications in gaming, training, design, and collaborative environments.

6.2. Final Remarks

The experiment demonstrated that the implementation of a distributed multi-broker system on the edge effectively mitigates congestion and reduces latency in managing large-scale data networks, surpassing the performance of a single-edge broker. The distributed multi-broker architecture enables closer and parallel connections with clients, facilitating faster processing of substantial data quantities. Consequently, network latency is significantly reduced, leading to improved overall system performance.

6.3. Future Research Directions

The findings of this experiment draw attention to the importance of considering efficient hardware resources on the edge when deploying edge systems, especially given the growing demand for edge computing. Future research can delve deeper into this aspect by exploring hardware resource optimization on the edge. This exploration may involve investigating methods to maximize computational power, storage capacity, and network bandwidth to further enhance the performance and capabilities of edge systems. By fully leveraging hardware resources, researchers can unlock new possibilities for deploying sophisticated applications and services on the edge.

Another promising direction for future work is the development of mechanisms for efficient data distribution on the edge. This entails exploring strategies for the parallel processing of large data volumes using the MQTT protocol with multi-brokers. By distributing the processing load across multiple brokers on the edge, better scalability and improved performance can be achieved. Additionally, future research could explore dynamic client migration techniques(specifically to ensure the state preservation of MQTT. This aspect was partially explored as a testcase during the experiment in this thesis, with further details provided in Appendix (0.1), intelligently routing clients to the most suitable and computationally optimized brokers based on their application requirements. This approach would enable efficient resource utilization and enhance the overall responsiveness and reliability of edge systems.

7. REFERENCES

[1] Mahmood S. (2021) Review of internet of things in different sectors: Recent advances, technologies, and challenges. Journal on Internet of Things .

[2] Chaudet C., Demeure I.M. & Ktari S. (2012) A model to evaluate brokers overlays for publish/subscribe in wireless sensor networks. 2012 9th Annual Conference on Wireless On-Demand Network Systems and Services (WONS) , pp. 71–74.

[3] Bhaddurgatte R.C. & Bp V.K. (2016) A review: Qos architecture and implementations in iot environment. Research & Reviews: Journal of Engineering and Technology 2016, pp. 6–12.

[4] Atlam H.F., Walters R.J. & Wills G.B. (2018) Fog computing and the internet of things: A review. Big Data Cogn. Comput. 2, p. 10.

[5] Satyanarayanan M. (2017) Edge computing. Computer 50, pp. 36–38.

[6] Dolui K. & Datta S.K. (2017) Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. 2017 Global Internet of Things Summit (GIoTS) , pp. 1–6.

[7] Mehrabi M., You D., Latzko V., Salah H., Reisslein M. & Fitzek F.H.P. (2019) Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey. IEEE Access 7, pp. 166079–166108.

[8] Liu G.Y. & Maguire G.Q. (1995) A predictive mobility management algorithm for wireless mobile computing and communications. Proceedings of ICUPC '95 - 4th IEEE International Conference on Universal Personal Communications , pp. 268–272.

[9] Rose K., Eldridge S. & Chapin L. (2015) The internet of things: An overview. The internet society (ISOC) 80, pp. 1–50.

[10] Ramson S.J., Vishnu S. & Shanmugam M. (2020) Applications of internet of things (iot)–an overview. In: 2020 5th international conference on devices, circuits and systems (ICDCS), IEEE, pp. 92–95.

[11] Khan M.N.R., Haque H., Labeeb K., Aktar M., Datta R.K. & Abedin M.Z. (2021) Internet of things and wireless sensor network solution in smart environmental monitoring. 2021 6th International Conference on Communication and Electronics Systems (ICCES) , pp. 1–5.

[12] Guner A., Kurtel K. & Celikkan U. (2017) A message broker based architecture for context aware iot application development. 2017 International Conference on Computer Science and Engineering (UBMK) , pp. 233–238.

[13] Görmez Y., Arslan H. & Kelek Ö.F. (2020) Efficient and scalable broker design for the internet of things environments. 2020 28th Signal Processing and Communications Applications Conference (SIU) , pp. 1–4.

[14] Goth G. (1999) Mobile devices present integration challenges. IT Professional 1, pp. 11–15.

[15] Chuang M.C., Ke S.A. & Chen C. (2020) Network controlled handover mechanisms in mobile edge computing. 2020 International Conference on Information and Communication Technology Convergence (ICTC) , pp. 756–761.

[16] Zhang Y., Groves T.L., Cook B., Wright N.J. & Coskun A.K. (2020) Quantifying the impact of network congestion on application performance and network metrics. 2020 IEEE International Conference on Cluster Computing (CLUSTER) , pp. 162–168.

[17] Yang Y., Wu L., Yin G., Li L. & Zhao H. (2017) A survey on security and privacy issues in internet-of-things. IEEE Internet of Things Journal 4, pp. 1250–1258.

[18] Ma Z., Xiao M., Xiao Y., Pang Z., Poor H.V. & Vucetic B. (2019) High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals, and enabling technologies. IEEE Internet of Things Journal 6, pp. 7946–7970.

[19] Silverio-Fernández M., Renukappa S. & Suresh S. (2018) What is a smart device?-a conceptualisation within the paradigm of the internet of things. Visualization in Engineering 6, pp. 1–10.

[20] Ryan P.J. & Watson R.B. (2017) Research challenges for the internet of things: what role can or play? Systems 5, p. 24.

[21] Rahmani A.M., Bayramov S. & Kiani Kalejahi B. (2022) Internet of things applications: opportunities and threats. Wireless Personal Communications 122, pp. 451–476.

[22] Badr Y., Zhu X. & Alraja M.N. (2021) Security and privacy in the internet of things: threats and challenges. Service Oriented Computing and Applications 15, pp. 257–271.

[23] Thangaramya K., Kulothungan K., Logambigai R., Selvi M., Ganapathy S. & Kannan A. (2019) Energy aware cluster and neuro-fuzzy based routing algorithm for wireless sensor networks in iot. Computer Networks 151, pp. 211–223.

[24] Wu F., Li X., Xu L., Vijayakumar P. & Kumar N. (2020) A novel three-factor authentication protocol for wireless sensor networks with iot notion. IEEE Systems Journal 15, pp. 1120–1129.

[25] Vaiyapuri T., Parvathy V.S., Manikandan V., Krishnaraj N., Gupta D. & Shankar K. (2021) A novel hybrid optimization for cluster-based routing protocol in information-centric wireless sensor networks for iot based mobile edge computing. Wireless Personal Communications , pp. 1–24.

[26] Jaiswal K. & Anand V. (2020) Eomr: An energy-efficient optimal multi-path routing protocol to improve qos in wireless sensor network for iot applications. Wireless Personal Communications 111, pp. 2493–2515.

[27] Kuthadi V.M., Selvaraj R., Baskar S., Shakeel P.M. & Ranjan A. (2022) Optimized energy management model on data distributing framework of wireless sensor network in iot system. Wireless Personal Communications 127, pp. 1377–1403.

[28] Skiadopoulos K., Tsipis A., Giannakis K., Koufoudakis G., Christopoulou E., Oikonomou K., Kormentzas G. & Stavrakakis I. (2019) Synchronization of data measurements in wireless sensor networks for iot applications. Ad Hoc Networks 89, pp. 47–57.

[29] Lazidis A., Tsakos K. & Petrakis E.G. (2022) Publish–subscribe approaches for the iot and the cloud: Functional and performance evaluation of open-source systems. Internet of Things 19, p. 100538.

[30] Blazy O., Conchon E., Klingler M. & Sauveron D. (2021) An iot attribute-based security framework for topic-based publish/subscribe systems. IEEE Access 9, pp. 19066–19077.

[31] Nguyen H., Uddin M.Y.S. & Venkatasubramanian N. (2019) Multistage adaptive load balancing for big active data publish subscribe systems. In: Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems, pp. 43–54.

[32] Tariq M.A., Koldehofe B. & Rothermel K. (2013) Securing broker-less publish/subscribe systems using identity-based encryption. IEEE transactions on parallel and distributed systems 25, pp. 518–528.

[33] Pavlopoulou N. & Curry E. (2021) Iotsax: A dynamic abstractive entity summarization approach with approximation and embedding-based reasoning rules in publish/subscribe systems. IEEE Internet of Things Journal 9, pp. 1830–1847.

[34] Javed A., Larijani H. & Wixted A. (2018) Improving energy consumption of a commercial building with iot and machine learning. IT Professional 20, pp. 30–38.

[35] Gündoğan C., Kietzmann P., Schmidt T.C. & Wählisch M. (2022) A mobility-compliant publish–subscribe system for an information-centric internet of things. Computer Networks 203, p. 108656.

[36] Zaarour T., Bhattacharya A. & Curry E. (2022) Openpubsub: supporting large semantic content spaces in peer-to-peer publish/subscribe systems for the internet of multimedia things. IEEE Internet of Things Journal 9, pp. 17640–17659.

[37] Ramachandran G.S., Wright K.L., Zheng L., Navaney P., Naveed M., Krishnamachari B. & Dhaliwal J. (2019) Trinity: A byzantine fault-tolerant distributed publish-subscribe system with immutable blockchain-based persistence. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, pp. 227–235.

[38] Pavlopoulou N. & Curry E. (2022) Possum: An entity-centric publish/subscribe system for diverse summarization in internet of things. ACM Transactions on Internet Technology (TOIT) 22, pp. 1–30.

[39] Nguyen L.T.T., Ha S.X., Le T.H., Luong H.H., Vo K.H., Nguyen K.H.T., Dao T.A., Nguyen H.V.K. et al. (2022) Bmdd: a novel approach for iot platform (broker-less and microservice architecture, decentralized identity, and dynamic transmission messages). PeerJ Computer Science 8, p. e950.

[40] Singh M., Rajan M., Shivraj V. & Balamuralidhar P. (2015) Secure mqtt for internet of things (iot). In: 2015 fifth international conference on communication systems and network technologies, IEEE, pp. 746–751.

[41] Thean Z.Y., Yap V.V. & Teh P.C. (2019) Container-based mqtt broker cluster for edge computing. In: 2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE), IEEE, pp. 1–6.

[42] Pranata A.A., Lee J.M. & Kim D.S. (2017) Towards an iot-based water quality monitoring system with brokerless pub/sub architecture. In: 2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), IEEE, pp. 1–6.

[43] Tabatabai S., Mohammed I., Al-Fuqaha A. & Salahuddin M.A. (2017) Managing a cluster of iot brokers in support of smart city applications. In: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE, pp. 1–6.

[44] Aburukba R.O., AliKarrar M., Landolsi T. & El-Fakih K. (2020) Scheduling internet of things requests to minimize latency in hybrid fog–cloud computing. Future Generation Computer Systems 111, pp. 539–551.

[45] Cui L., Xu C., Yang S., Huang J.Z., Li J., Wang X., Ming Z. & Lu N. (2018) Joint optimization of energy consumption and latency in mobile edge computing for internet of things. IEEE Internet of Things Journal 6, pp. 4791–4803.

[46] Shukla S., Hassan M.F., Khan M.K., Jung L.T. & Awang A. (2019) An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment. PloS one 14, p. e0224934.

[47] Alshammari H.H. (2023) The internet of things healthcare monitoring system based on mqtt protocol. Alexandria Engineering Journal 69, pp. 275–287.

[48] Li J., Zhang T., Jin J., Yang Y., Yuan D. & Gao L. (2017) Latency estimation for fog-based internet of things. In: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, pp. 1–6.

[49] Shukla S., Hassan M.F., Tran D.C., Akbar R., Paputungan I.V. & Khan M.K. (2021) Improving latency in internet-of-things and cloud computing for real-time data transmission: a systematic literature review (slr). Cluster Computing , pp. 1–24.

[50] Wang Z., Gao F. & Jin X. (2020) Optimal deployment of cloudlets based on cost and latency in internet of things networks. Wireless Networks 26, pp. 6077–6093.

[51] Velasquez K., Abreu D.P., Curado M. & Monteiro E. (2017) Service placement for latency reduction in the internet of things. Annals of Telecommunications 72, pp. 105–115.

[52] Aazam M. & Huh E.N. (2015) Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In: 2015 ieee 29th international conference on advanced information networking and applications, IEEE, pp. 687–694.

[53] Purwania I.B.G., Kumara I.N.S. & Sudarma M. (2020) Application of iot-based system for monitoring energy consumption. International Journal of Engineering and Emerging Technology 5, pp. 81–93.

[54] Tekin N., Acar A., Aris A., Uluagac A.S. & Gungor V.C. (2023) Energy consumption of on-device machine learning models for iot intrusion detection. Internet of Things 21, p. 100670.

[55] Mocnej J., Miškuf M., Papcun P. & Zolotová I. (2018) Impact of edge computing paradigm on energy consumption in iot. IFAC-PapersOnLine 51, pp. 162–167.

[56] Alkhayyat A., Thabit A.A., Al-Mayali F.A. & Abbasi Q.H. (2019) Wbsn in iot health-based application: toward delay and energy consumption minimization. Journal of Sensors 2019.

[57] Rausch T., Dustdar S. & Ranjan R. (2018) Osmotic message-oriented middleware for the internet of things. IEEE Cloud Computing 5, pp. 17–25.

[58] Roy D.G., Mahato B., De D. & Buyya R. (2018) Application-aware end-to-end delay and message loss estimation in internet of things (iot)—mqtt-sn protocols. Future Generation Computer Systems 89, pp. 300–316.

[59] Khare A., Sharma R. & Ahuja N.J. (2020) Experimental investigation of integrated id method to mitigate message loss in iot control devices. Journal of Engineering Science and Technology (JESTEC) 15, pp. 32–45.

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

[61] Kavuri S., Moltchanov D., Ometov A., Andreev S. & Koucheryavy Y. (2020) Performance analysis of onshore nb-iot for container tracking during near-the-shore vessel navigation. IEEE Internet of Things Journal 7, pp. 2928–2943.

[62] Schmitt A., Carlier F. & Renault V. (2018) Dynamic bridge generation for iot data exchange via the mqtt protocol. Procedia computer science 130, pp. 90–97.

[63] Koziolek H., Grüner S. & Rückert J. (2020) A comparison of mqtt brokers for distributed iot edge computing. In: Software Architecture: 14th European Conference, ECSA 2020, L'Aquila, Italy, September 14–18, 2020, Proceedings 14, Springer, pp. 352–368.

[64] Luzuriaga J.E., Perez M., Boronat P., Cano J.C., Calafate C. & Manzoni P. (2015) A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. In: 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), IEEE, pp. 931–936.

[65] Kawaguchi R. & Bandai M. (2019) A distributed mqtt broker system for location-based iot applications. 2019 IEEE International Conference on Consumer Electronics (ICCE) , pp. 1–4.

[66] Mishra B., Mishra B. & Kertesz A. (2021) Stress-testing mqtt brokers: A comparative analysis of performance measurements. Energies 14, p. 5817.

[67] Pourreza M. & Narasimhan P. (2022) Quantitative evaluation of application layer protocols for edge-centric real-time iovt applications. In: 2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, pp. 0067–0075.

[68] Longo E., Redondi A.E., Cesana M., Arcia-Moret A. & Manzoni P. (2020) Mqtt-st: a spanning tree protocol for distributed mqtt brokers. In: ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE, pp. 1–6.

[69] Mukherjee A., Dey N. & De D. (2020) Edgedrone: Qos aware mqtt middleware for mobile edge computing in opportunistic internet of drone things. Computer Communications 152, pp. 93–108.

[70] Sommer P. & Wattenhofer R. (2009) Gradient clock synchronization in wireless sensor networks. In: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, IPSN '09, IEEE Computer Society, USA, p. 37–48.

8. APPENDICES

0.1. Migration of Microservices - Broker Container State Preservation

0.1.1. Introduction

In this appendix, we provide additional details on the research conducted to achieve the state preservation of a broker container with MQTT configuration during migration. The objective of this research was to develop a method that allows seamless transfer of the container's state to other systems, ensuring a smooth resumption of operations from where it left off.

0.1.2. Methodology

To achieve the state preservation of the broker container, the following steps were undertaken:

Container Configuration

The broker container was initially configured with the necessary MQTT settings, including topic subscriptions, access controls, and persistent storage configurations. This ensured that the container operated with the required MQTT parameters and retained its state throughout the migration process.

Exportability

To make the container exportable to other systems, a comprehensive export mechanism was devised. This mechanism packaged the container, along with its MQTT configuration and associated state data, into a portable format. This allowed for easy transfer and deployment onto target systems while maintaining the container's integrity and state.

0.1.3. Results

The research yielded successful outcomes, as demonstrated by the following key results:

State Preservation

The broker container, after being exported from the source system, was seamlessly deployed onto a new system. It retained its MQTT configuration and state, including active connections, retained messages, and session data. This ensured a smooth resumption of operations without any loss of critical information or disruption in service.

Compatibility

The exported container was successfully deployed onto various target systems with different underlying infrastructures, such as cloud platforms or on-premises environments. The compatibility of the container's state preservation mechanism across these diverse systems highlights its versatility and adaptability.

0.1.4. Conclusion

In conclusion, the research on the migration of microservices, specifically focusing on the preservation of a broker container's state with MQTT configuration, has demonstrated the feasibility and effectiveness of the proposed method. By ensuring the seamless transfer and resumption of the container's state, this approach will offer significant benefits.

Keywords: Micro-services, edge-cloud, edge services, migration, state-maintained docker, QoS.