Queuing Theory-Based Modeling and Optimization of a Publish/Subscribe IoT Communication System

Franc Pouhela[1], Anthony Kiggundu[1], Hans D. Schotten[12]
[1]*German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern*
Email: {franc.pouhela; anthony.kiggundu; hans_dieter.schotten}@dfki.de
[2]*University of Kaiserslautern (RPTU), Germany*
Email: {schotten}@rptu.de

***Abstract*—This paper presents a comprehensive Queuing Theory-based model of a Publish/Subscribe Internet of Things (IoT) communication system. Using a Markovian (`M/M/1`) queuing process, we model, analyze, and optimize the message delivery of a Middleware Message Queuing Protocol (MMQP)-broker. We derive key performance metrics such as average waiting time, system utilization, and the effects of multi-threading on performance. Our model aims to optimize the throughput and the efficiency of the system by determining the optimal service rate of the broker in relation to the overall arrival rate of messages. Through experimental evaluation, we demonstrate the model's accuracy and its relevance in providing valuable insights for improving IoT communication.**

***Index Terms*—IoT, Queuing Theory, Modeling, 6G**

I. Introduction

The growing demand for efficient, scalable, and continuous data collection and processing in distributed systems has received significant attention over the past decade, especially in the context of next-generation mobile networks such as 5G and the anticipated 6G. A major factor driving this trend is the extensive desire to achieve context-aware Internet of Things (IoT) communication by integrating Artificial Intelligence (AI) in different layers of network operations.

A promising avenue is found in a Context Management System (CoMaS) [1], [2]. A CoMaS is a framework that leverages network context data and applies various AI algorithms to gain deeper insights into the network's state. This knowledge is then utilized to optimize and enhance network operations. The success of such a system hinges on the efficient and seamless acquisition and distribution of contextual data, a process that becomes increasingly challenging with the exponential growth of connected devices, as predicted in IoT.

The Message Queuing Telemetry Transport (MQTT) [3] protocol, widely recognized as a lightweight messaging protocol, is extensively used in IoT and Machine-to-Machine (M2M) scenarios where scalability and efficiency are paramount. However, upon a closer examination of MQTT, we identified several areas with room for improvement. To address the limitations of MQTT, we designed a similar protocol called: Middleware Message Queuing Protocol (MMQP) [4] that enhances efficiency, flexibility, scalability, and security.

To ensure optimal performance and reliability of the broker under varying network conditions, it was crucial to mathematically model its behavior. Queuing theory offers a robust framework for analyzing and predicting performance by modeling message flow through the system. This approach enables the evaluation of key performance metrics such as latency, queue size, system utilization, etc., which are essential for assessing the efficiency and scalability of our framework.

The remainder of this paper is organized as follows: Section II reviews the related work. Section III provides a concise introduction to the MMQP protocol. Section IV presents our queuing model, and Section V offers a comprehensive evaluation of this model. Finally, Section VI concludes the study and outlines potential directions for future research.

II. Related Work

The Publish/Subscribe communication paradigm has been widely studied, particularly in the context of distributed systems and IoT applications. MQTT [3] and Advanced Message Queuing Protocol (AMQP) [5] are prominent implementations of this paradigm, with similar mechanisms for message distribution and Quality of Service (QoS) management.

Queuing theory has been extensively applied to model and optimize various aspects of communication systems. Kleinrock [6] pioneered the use of queuing models to analyze network performance, laying the foundation for subsequent work in the field. More recently, authors in [7], [8] have utilized queuing models to analyze the performance of wireless and mobile networks. These studies provide a robust framework for understanding system behavior under different traffic conditions.

Eugster et al. [9] provided a comprehensive survey of the publish/subscribe paradigm, outlining various design choices and their implications for system performance. Specific to MQTT, authors in [10], [11] have examined the protocol's design and implementation, highlighting its suitability for resource-constrained environments such as IoT.

Several works have explored optimization techniques in the context of IoT communication. Bonomi et al. [12] introduced the concept of Fog Computing, which optimizes resource allocation in IoT networks through localized processing.

While significant progress has been made in both queuing theory and publish/subscribe systems, there is a notable gap in research that integrates these fields for system optimization.

III. Communication System

Efficient and secure IoT communication depends heavily on the choice of messaging pattern and protocol. The publish/subscribe messaging pattern, exemplified by the widely adopted MQTT[3] protocol, has become the standard for IoT communication. In this study, we introduce a novel protocol named MMQP[4], inspired by MQTT. MMQP is a lightweight, binary protocol specifically designed for IoT and M2M communication. MMQP allows clients to communicate indirectly via a central distribution node known as the broker. Clients can act as both publishers and subscribers. Subscribers submit interest in specific topics by subscribing to them and publishers publish messages to these topics, which are named channels that categorize the message content. The broker distributes messages from publishers to the appropriate subscribers by managing topic hierarchies.

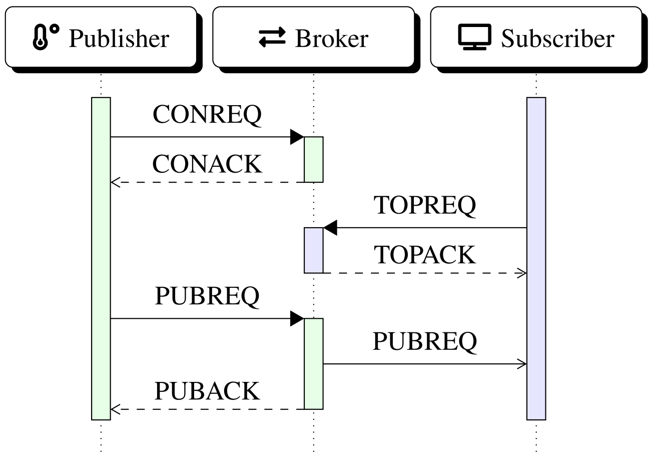

Fig. 1: MMQP Sequence Diagram

As illustrated in the sequence diagram depicted in Figure 1, the publisher establishes a connection with the broker by sending a CONREQ packet, to which the broker responds with a CONACK packet acknowledging the successful connection. Subsequently, the subscriber subscribes to a specific topic using the TOPREQ packet. Meanwhile, the publisher sends a PUBREQ (publish request) packet to the broker, which can be queued for further processing depending on the QoS level of the message. The broker then forwards the relevant message to the subscribers.

A. Message Routing

Topics in MMQP function as key identifiers for organizing and distributing messages. These topics are managed hierarchically by brokers and are represented as strings with multiple levels separated by forward slashes ("/"). When a client publishes a message, the broker analyzes the topic hierarchy to identify the appropriate subscribers. The broker then forwards the message to subscribers who are interested in the exact topic or any of its parent levels. For example, publishing a message under *home/room1/light* would prompt the broker to deliver the message not only to subscribers of *home/room1/light* but also to those subscribed to *home/room1* and *home*. See [4] for a thorough description.

B. Quality of Service

MMQP supports three levels of QoS. Messages published to the broker with a QoS level greater than 0 can be queued until their expiry interval is reached. If new clients join, the broker will forward the queued messages accordingly to their subscription pattern.

QoS 0 (At most once): Or best effort delivery, ensures the delivery without any acknowledgment (Figure 2). If the broker is currently busy, the packet may be dropped. This QoS level is primarily used in high data rate scenarios, such as video streaming, where occasional packet loss is not critical.

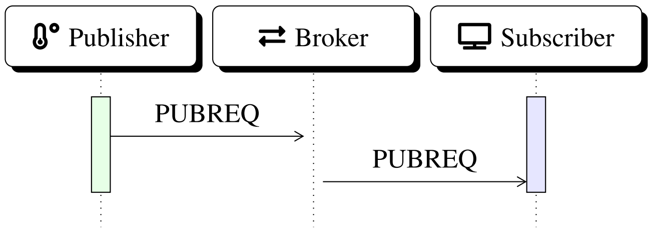

Fig. 2: Publishing with QoS level 0

QoS 1 (At least once): Here, the sender publishes the message, and the receiver (broker or subscriber) acknowledges its receipt. If the acknowledgment is not received, the publisher may resend the message (see Figure 3).

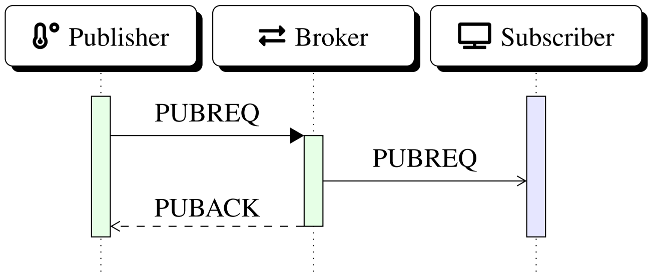

Fig. 3: Publishing with QoS level 1

QoS 2 (Exactly once): This QoS level guarantees that the message is delivered exactly once. It involves a two-step process as depicted in (Figure 4).

In the sequence diagrams shown in Figures 2,3, and 4, it is assumed that the subscriber receives messages with QoS level 0. Consequently, no acknowledgment is sent from the subscriber to the broker. This assumption was made to optimize the use of space available in the paper.

IV. System Modeling

As introduced in (Section I), queuing theory is a mathematical study of waiting lines, or queues, which is used to model the behavior of systems that provide services to randomly arriving customers. It is a fundamental aspect of operations

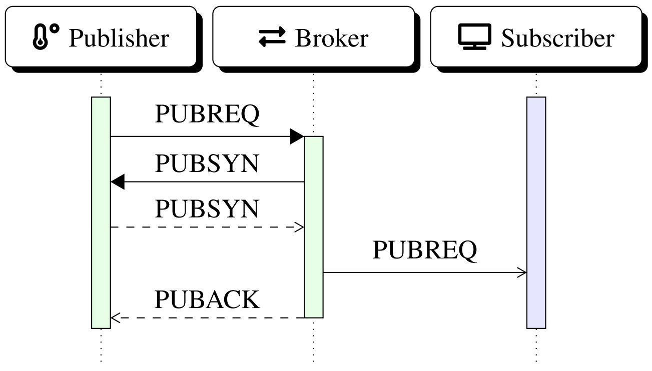

Fig. 4: Publishing with QoS level 2

research and is widely applied in various fields, including telecommunications, computer networks, manufacturing, and service systems.

The origins of queuing theory can be traced back to the work of A.K. Erlang in the early 20th century, who developed models to describe the congestion and waiting times in telephone networks [13]. Since then, queuing theory has evolved significantly, and its application has expanded to a wide range of domains. The theory provides essential tools for analyzing systems where there is a need to understand and predict system behavior under different load conditions, which is crucial for optimizing reliability and performance.

A typical queuing system is characterized by the arrival process of customers, the service process, and the queue discipline, which determines the order in which customers are served [14]. The simplest and most studied model is the `M/M/1` (where the "M" in the first and second octets denote Markovian arrival and service schemes respectively for a single queue (third octet). The arrivals follow a Poisson process, service times are exponentially distributed, and there is a single server [6]. This basic model forms the foundation for more complex queuing systems.

A. System Dynamics

To effectively model our communication system, it is essential to establish several key assumptions to simplify the complexity of the system while ensuring that the model accurately reflects its behavior. Following are the assumptions made as well as the state variables representing the system:

- First-In-First-Out (FIFO) queue discipline.
- Messages are reliably delivered to subscribers.
- The broker operates as a single-server queue.
- The system is assumed to be in a steady-state, allowing the of steady-state probability distributions.
- K and S represent the set of publishers and subscribers.
- $\mu_s \geqslant 0$ is the service rate of the subscriber $s \in S$.
- $\lambda_k \geqslant 0$ is the arrival rate of the publisher $k \in K$.

B. Overall Arrival Rate

Figure 5 illustrates the queuing model of our communication system. On the left is the set of publishers K characterized by their stochastic message arrival rate $\lambda_1, \ldots, \lambda_k$. On the right is the set of subscribers S, each associated with its respective and independent service rate $\mu_1, \ldots, \mu_s$ and in the middle is the broker acting as a service node in a `M/M/1` queue model. Each publisher k broadcasts messages at an independent rate λ_k, which follows a Poisson distribution. This is described in Equation 1, where $M_k(t) = m$ represents the number of messages sent by the publisher k in the time interval t.

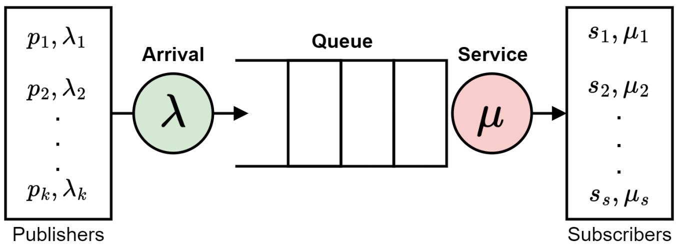

Fig. 5: System Queuing Model

$$P_k(M_k(t) = m) = \frac{e^{-\lambda_k t}(\lambda_k t)^m}{m!} \tag{1}$$

The overall arrival rate of the system can be described as the sum of the arrival rates of all publishers:

$$\lambda = \sum_{k=1}^{K} \lambda_k$$

This also results in Poisson distribution as characterized in Equation 2, where K represents the total number of publishers, $N(t)$ represents the total number of messages sent by all publishers combined in the time interval t.

$$P(N(t) = n) = \sum_{k=1}^{K} P_k(M_k(t) = m_k) = \frac{e^{-\lambda t}(\lambda t)^n}{n!} \tag{2}$$

The overall arrival rate effectively combines the independent rates of all publishers, allowing us to model the total message flow as a single Poisson process that reflects the stochastic nature of message arrivals.

C. Overall Service Rate

We have a single broker that forwards messages to a set of subscribers S. Each subscriber s has an exponentially distributed service time characterized by its service rate μ_s. The service time T_s for each subscriber follows the distribution:

$$F_s(T_s \leqslant t) = e^{-\mu_s t}$$

where μ_s is the service rate (or the mean rate of message processing) for subscriber s. This exponential distribution implies that the probability of a message being processed by time t increases with t, but the rate of increase slows down as t grows, which reflects the memory-less property of exponential distributions. The overall effective system's service rate μ can be defined as:

$$\mu = \sum_{s=1}^{S} \mu_s$$

This combined service rate μ indicates the expected rate at which messages are processed collectively across all subscribers. This summation is valid under the assumption that each subscriber is independent and that the system operates in a load-balanced manner with messages distributed uniformly across subscribers. The service completion time for S subscribers, where each subscriber's service time is defined by an exponential distribution with rate μ_s, follows an Erlang distribution given by:

$$F(T \leqslant t) = \frac{\mu_s e^{-\mu_s t}(\mu_s t)^{S-1}}{\Gamma(S)} \tag{3}$$

where $\Gamma(S)$ is the Gamma function, often expressed as $\Gamma(S) = (S-1)!$ for integer values of S. This formula describes the probability density function of the time it takes for a message to be fully processed by S subscribers, accounting for the multi-subscriber, concurrent processing model. Because each service time T_s is exponentially distributed, the expected service time for each subscriber can be represented as:

$$T_s = 1/\mu_s$$

D. Steady-State Indicators

The traffic intensity ρ is defined as: $\rho = \lambda/\mu$. It helps determine the system's load. When $\rho < 1$, the system is stable, meaning the broker can handle incoming messages without indefinitely growing the queue. If $\rho \geqslant 1$, the system is unstable, with the queue length potentially growing unbounded.

The probability that there are n messages in the system (being processed or in the queue) in steady-state is given by:

$$P(n) = (1-\rho)\rho^n$$

This formula is valid for $\rho < 1$ and describes the probability of the system having n messages waiting or being processed. The average number of messages in the system is:

$$L = \frac{\rho}{1-\rho}$$

The average time a message spends in the system (both waiting and processing time) is:

$$W = \frac{1}{\mu - \lambda}$$

The average time a message spends waiting in the queue (before being processed by the broker) is defined as:

$$W_q = \frac{\rho}{\mu-\lambda} = \frac{\lambda}{\mu(\mu-\lambda)}$$

E. Optimization Constraint

To optimize our communication system, we can formulate an optimization problem that minimizes latency (average waiting time) while ensuring system stability and maximizing throughput. The key goal here is to optimize the broker's service rate (μ) of messages while keeping the waiting time within acceptable limits. However, a growing service rate μ usually comes with additional operational cost (e.g., more processing power, higher network bandwidth, higher memory usage, etc.). So, we need to incorporate a cost term associated with the service rate growth in the optimization objective.

a) Stability Constraint: $\rho = \lambda/\mu < 1 \implies \mu > \lambda$

b) Service Rate Bound: Hardware limitations $\mu \leqslant \mu_{\text{max}}$

c) Cost Constraint: In our previous study described in [15], our measurements demonstrated that the power consumption of the broker exhibited a linear increase with growing traffic intensity. Based on this observation, let $C(\mu, t)$ represent the cost of maintaining a service rate μ, we define the cost function as:

$$C(\mu, t) = c_0 + c_1(t) \cdot \mu$$

where c_0 is the base cost and $c_1(t)$ is per unit increase in μ. Assuming the worst-case scenario where every incoming message from publisher $k \in K$ has to be forwarded to every subscribers $s \in S$, we could define $c_1(t)$ as:

$$c_1(t) = \frac{c_0}{\lambda(t) \cdot |S|} \implies C(\mu, \lambda) = c_0 + \frac{c_0}{\lambda \cdot |S|} \cdot \mu \tag{4}$$

The cost constraint can be define in relation to μ and λ as:

$$\boxed{C(\mu, \lambda) = c_0 + \frac{c_0}{\lambda \cdot |S|} \cdot \mu} \tag{5}$$

F. Optimization Problem

The optimization problem is formulated in equation (6), where $W(\mu, \lambda)$ represents the latency, and $\epsilon \cdot C(\mu, \lambda)$ introduces a penalty for higher service rates. By adjusting the parameter ϵ, the system can be tuned to prioritize either reducing latency or minimizing operational costs (power consumption). This balance allows for flexibility in aligning system performance with specific application requirements, such as low-latency demands or cost-efficiency goals.

$$O(\mu, \lambda) = \min_{\mu} \quad [W(\mu, \lambda) + \epsilon \cdot C(\mu, \lambda)]$$

subject to:

$$\begin{aligned} &\epsilon > 0 \\ &\lambda \leqslant \lambda_{max}, \\ &\lambda < \mu \leqslant \mu_{max}, \\ &W(\mu, \lambda) = \frac{1}{\mu - \lambda}, \\ &C(\mu, \lambda) = c_0 + \frac{c_0}{\lambda \cdot |S|} \cdot \mu \end{aligned} \tag{6}$$

1) Objective Function: The objective function $\mathcal{F}(\mu, \lambda)$ depicted in (7) depends on both μ and λ. To find the optimal service rate μ in relation to the arrival rate λ, we can calculate the partial derivative of the objective function with respect to μ and equate it to zero to find the critical point.

$$\mathcal{F}(\mu, \lambda) = \frac{1}{\mu - \lambda} + \epsilon \cdot \left(c_0 + \frac{c_0}{\lambda \cdot |S|} \cdot \mu \right) \tag{7}$$

The partial derivative with respect to μ is:

$$\frac{\partial \mathcal{F}}{\partial \mu} = -\frac{1}{(\mu - \lambda)^2} + \frac{\epsilon c_0}{\lambda |S|} \tag{8}$$

Settings $\frac{\partial \mathcal{F}}{\partial \mu} = 0$, give us:

$$\mu^* = \lambda + \sqrt{\frac{\lambda \cdot |S|}{c_0 \cdot \epsilon}} \tag{9}$$

We can approximate c_0 by setting the optimal service rate to its minimum $\mu^*(\lambda) \approx 0$ and the penalty parameter to $\epsilon = 1$. This implies:

$$c_0 \approx \frac{|S|}{\lambda} \tag{10}$$

Inserting (10) into (9) gives us:

$$\boxed{\mu^*(\lambda, \epsilon) = \lambda \left(1 + \sqrt{\frac{1}{\epsilon}} \right) \quad \text{with} \quad \lambda < \mu, \quad \epsilon > 0} \tag{11}$$

This solution allows us to minimize costs while adhering to the constraints, ensuring that μ remains close to λ for stability and minimal additional expense. The penalty parameter ϵ helps prioritize latency over resource and vice-versa.

V. Experimental Evaluation

For our experimental evaluation, we utilized a custom MMQP-broker, with similar results expected from an MQTT broker. The broker is implemented in C++17, compatible with both Windows and Linux Operating Systems (OSs), and leverages the Asio library [16] for cross-platform, asynchronous network and low-level I/O operations. To ensure flexible and scalable behavior, the broker's software architecture is based on the Entity-Component-System (ECS) design pattern [17]. This architecture decouples object (entities) from their data (component) and behavior or logic (systems).

A. Measurement Setup

The dataset (Table I) used in this evaluation was collected on a laptop equipped with an *11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz processor and 32GB of RAM*. In our setup, publishers sent messages every `100ms` (`10Hz`) with payload size ranging from `10B` to `10K` to the broker, which forwards them to subscribers while measuring arrival and service rates of messages and bytes. Similar measurements was also performed with varying number of threads ranging from `1` to `8` (Sections: V-C) to evaluate the impact of multi-threading on system performance.

TABLE I: Datasets Snippet

clients	message (μ)	message (λ)	byte (μ)	byte (λ)
5	230.3	46.0	2360239.2	472047.8
10	914.2	91.4	9367771.4	936777.1
15	2063.5	137.5	21143352.8	1409556.8
20	3674.2	183.7	37646731.4	1882336.5
25	5732.1	229.2	58731535.7	2349261.4
30	8271.4	275.6	84749057.1	2824968.5
35	11252.4	321.4	115293115.0	3294089.0
40	14662.8	366.5	150235634.2	3755890.8

B. Measurements for n-Clients

This scenario evolves an equal number n of publishers and subscribers in single-threaded mode. The expectation in this case is that for m incoming messages, $n \cdot m$-messages will be processed and forwarded by the broker. Figure 6 a linear growth of the arrival rate which suggests that each additional client contributes an almost constant increase in the message load arriving at the broker. A similar pattern is observe with different payload sizes. This can also be represented as:

$$\lambda(n) = 9.17 * n + 0.055 \tag{12}$$

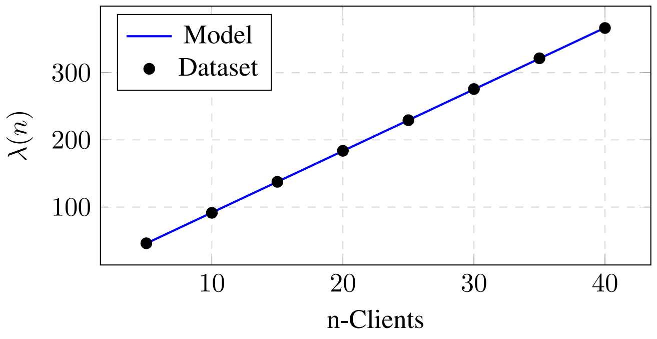

Fig. 6: Arrival rate growth with 10K payload

As the number of clients (`n`) increases, the service rate increases significantly (Figure 7). This makes sense since more clients connected to the broker should generate more messages, which the system has to process. The relationship between the number of clients `n` and μ appears to be quadratic or exponential. An approximated formulation for the growth of μ can be described as:

$$\begin{aligned} \mu_{quad}(n) &= 9.14n^2 + 1.62n - 12.12 \\ \mu_{exp}(n) &= 3317.75 \cdot e^{0.0435n} - 4121.82 \end{aligned} \tag{13}$$

The decreasing values of system utilization $\rho = \lambda/\mu$ in Figure 8 suggest that the broker's service capacity is scaling effectively, allowing the system to maintain low utilization, high responsiveness, and operational stability as traffic increases.

C. Multi-Threaded Measurement

This scenario involved testing the impact of multi-threading on the Key Performance Indicatorss (KPIs). By progressively

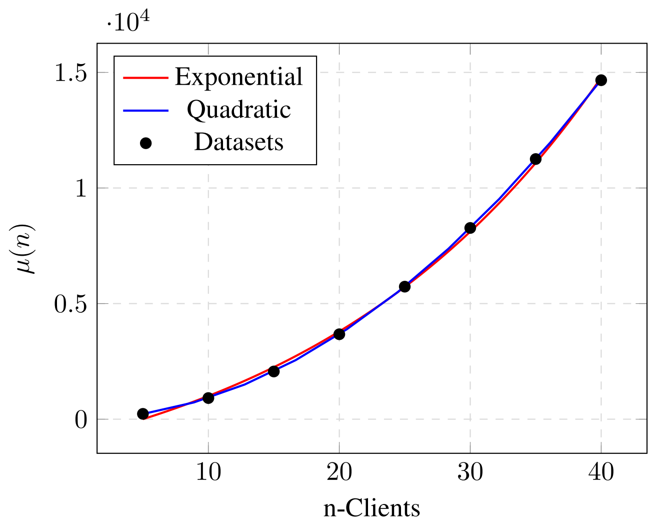

Fig. 7: Service rate growth with 10K payload size

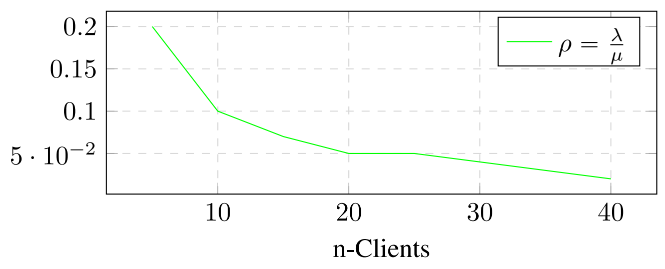

Fig. 8: System utilization with 10K payload size

increasing the number of worker-threads `w` with a fixed payload size and number of clients, we measured the arrival and service rates of the broker.

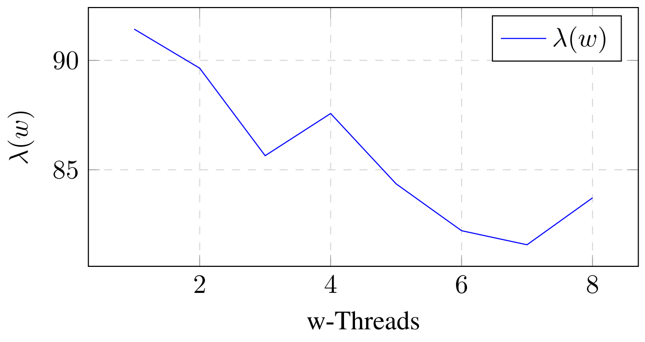

Fig. 9: Multi-threaded arrival rate for 10K payload size

The arrival rate λ (Figure 9) decreases slightly as the number of threads increase. It starts at around `91.43` for `1` thread and drops to `81.57` for `7` threads, before slightly increasing again to `83.71` for `8` threads. This slight reduction in λ can be caused by the context switch between threads. In addition to reading messages, the system now has to also manage concurrent threads.

The service rate μ (Figure 10) remains relatively stable, hovering around `914` messages per second, with slight fluctuations as the number of threads `w` increases. Increasing the number of threads has a minimal (not positive) impact on the overall service rate of the broker.

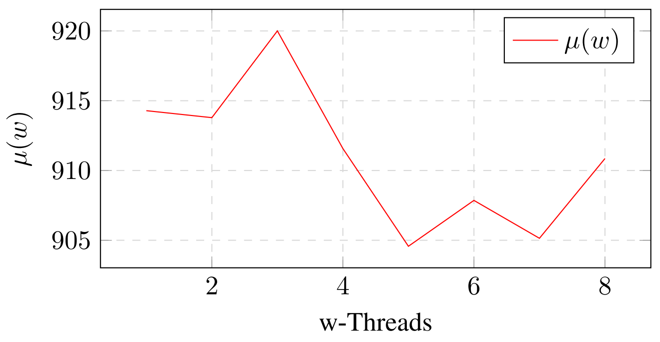

Fig. 10: Multi-threaded service rate for 10K payload size

D. Optimization Results

Table II provides some computed values based on the optimization model describe in Section IV. From this dataset, we can observe how the optimal service rate μ^* responds to variations in the penalty parameter ϵ and the arrival rate λ. As λ increases, we see (Figure 11) a clear, consistent rise in the optimal service rate μ^*. This relationship suggests that μ^* must scale with λ to maintain system stability, likely to ensure that the service rate remains sufficiently higher than the arrival rate, minimizing latency and keeping the system responsive.

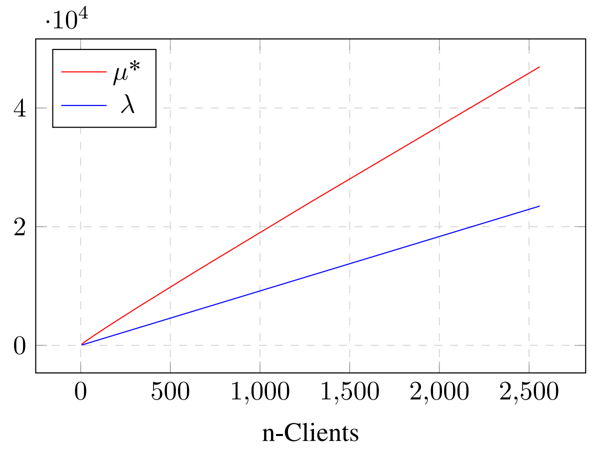

Fig. 11: Optimal service rate μ^* growth compared to arrival rate λ

Figure 12 shows that the optimal service rate μ^* grows significantly with increases in λ and ϵ, but it is more sensitive to changes in λ than in ϵ. This is advantageous, as it avoids excessive penalization of the system, reducing the need to drop messages excessively to maintain low operational costs.

VI. Conclusion and Future Work

In conclusion, this study demonstrates the potential benefits of applying queuing theory to model and optimize the performance of a publish/subscribe communication system. The findings from our experimental evaluation suggest that

TABLE II: Optimization Results

number_clients (n)	**service_rate** (μ)	**arrival_rate** (λ)	**penalty_parameter** (ϵ)	**cost_parameter** (c_0)	**optimal_service** (μ^*)
5	224.48	45.91	0.10	0.108920597	191.06
10	918.08	91.76	0.20	0.108985886	296.92
20	3676.28	183.46	0.30	0.109018560	518.39
40	14676.68	366.86	0.40	0.109034905	946.90
80	58613.48	733.66	0.50	0.109043079	1771.19
160	234231.08	1467.26	0.60	0.109047166	3361.47
320	936442.28	2934.46	0.70	0.109049210	6441.79
640	3744768.68	5868.86	0.80	0.109050232	12430.43
1280	14977037.48	11737.66	0.90	0.109050743	24110.22
2560	59904039.08	23475.26	1.00	0.109050999	46950.51

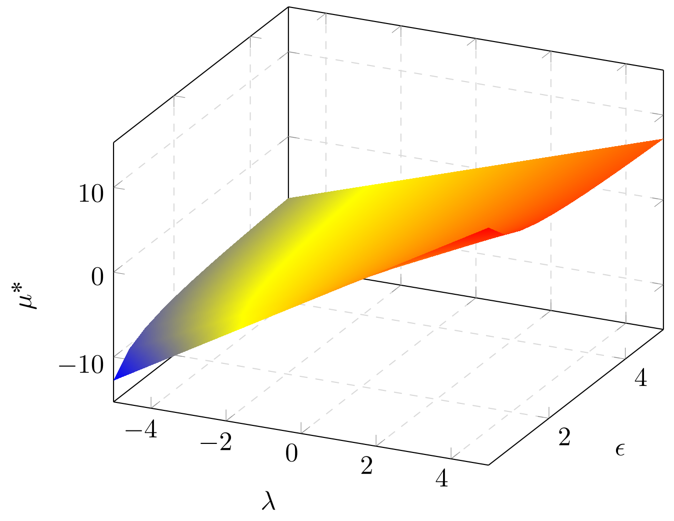

Fig. 12: Optimal service rate μ^* growth compared to arrival rate λ

the arrival rate λ primarily drives the growth of the optimal service rate μ^* to ensure stability, while the penalty parameter ϵ adjusts this growth by balancing between latency reduction and cost efficiency. The proposed optimization methodology provides a foundation for developing more efficient algorithms for managing message queues.

Future research should focus on using more real-world data to improve the optimization's accuracy. Additionally, implementing various Machine Learning (ML) algorithms to predict traffic growth and determine the optimal penalty parameter could further enhance the balance between latency and resource utilization.

Acknowledgment

The authors acknowledge the financial support by the German *Federal Ministry for Education and Research (BMBF)* within the projects Open6GHub {16KISK003K} & 6G-Terafactory {16KISK186}.

References

[1] Pouhela, F., Krummacker, D., and Schotten, H. D., "Towards 6G Networks," in *A Context Management Architecture for Decoupled Acquisition and Distribution of Information in Next-Generation Mobile Networks*, ser. ITG, vol. 157, VDE. IEEE, 5 2023. [Online]. Available: https://www.researchgate.net/publication/373328855_A_Context_Management_Architecture_for_Decoupled_Acquisition_and_Distribution_of_Information_in_Next-Generation_Mobile_Networks

[2] Pouhela, F., Sanon, S. P., and Schotten, H. D., "Ngna 2023," in *A Differential Privacy Approach for Context-Aware Service Provisioning in Mobile Networks*, 12 2023. [Online]. Available: https://www.researchgate.net/publication/377307205_A_Differential_Privacy_Approach_for_Context-Aware_Service_Provisioning_in_Mobile_Networks

[3] OASIS. Mqtt version 5.0, oasis standard. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[4] Pouhela, F., Sanon, S. P., Krummacker, D., and Schotten, H. D., "Everything Interconnected via Cyberspace," in *MMQP: A Lightweight, Secure and Scalable IoT Communication Protocol*, IEEE. IEEE, 8 2024. [Online]. Available: https://www.researchgate.net/publication/381742313_MMQP_A_Lightweight_Secure_and_Scalable_IoT_Communication_Protocol

[5] OASIS. Advanced message queuing protoco. [Online]. Available: https://www.amqp.org/

[6] Kleinrock, L., *Queueing Systems, Volume 1: Theory*. Wiley-Interscience, 1975.

[7] Singh, J. P. and Dutta, P., "Temporal behavior analysis of mobile ad hoc network with different mobility patterns," in *Proceedings of the international Conference on Advances in Computing, Communication and Control*, 2009, pp. 696–702.

[8] Bisnik, N. and Abouzeid, A., "Queuing network models for delay analysis of multihop wireless ad hoc networks," in *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, 2006, pp. 773–778.

[9] Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M., "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[10] Hunkeler, U., Truong, H. L., and Stanford-Clark, A., "Mqtt-s—a publish/subscribe protocol for wireless sensor networks," in *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*. IEEE, 2008, pp. 791–798.

[11] Light, R. A., "Mosquitto: server and client implementation of the mqtt protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.

[12] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S., "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[13] Erlang, A., "The theory of probabilities and telephone conversations," *Nyt Tidsskrift for Matematik B*, vol. 20, pp. 33–39, 1909.

[14] Gross, D., Shortle, J. F., Thompson, J. M., and Harris, C. M., *Fundamentals of Queueing Theory*. John Wiley & Sons, 2008.

[15] Pouhela, F., Arabshahi, M., and Schotten, H. D., "Everything Interconnected via Cyberspace," in *Analyzing and Predicting the Power Consumption of a Publish/Subscribe IoT-Broker*, IEEE. IEEE, 8 2024. [Online]. Available: https://www.researchgate.net/publication/381742346_Analyzing_and_Predicting_the_Power_Consumption_of_a_PublishSubscribe_IoT-Broker

[16] Asio c++ library. [Online]. Available: https://think-async.com/Asio/

[17] Pouhela, F., Krummacker, D., and Schotten, H. D., "Entity component system architecture for scalable, modular, and power-efficient iot-brokers," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–6. [Online]. Available: https://www.researchgate.net/publication/373318000_Entity_Component_System_Architecture_for_Scalable_Modular_and_Power-Efficient_IoT-Brokers